# All Terrain Semi-Autonomous Rover with Target Acquisition and GPS Tracking

| Sl. No. | Reg. No. | Student Name |
|---|---|---|
| 1 | 16ETEC004401 | Annoi Roy |
| 2 | 16ETEC004028 | Kiran Prabhakar |
| 3 | 16ETEC004046 | Nikhil C |
| 4 | 16ETEC004032 | M Rahul Naidu |

**Supervisor(s):**Mrs. Vasanthavalli S.

August– 2020

B. Tech. in Electronics and CommunicationEngineering

FACULTY OF ENGINEERING AND TECHNOLOGY
M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES
Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

# *Certificate*

*This is to certify that the Project titled "All Terrain Semi-Autonomous Rover with Target Acquisition and GPS Tracking" is a bonafide work carried out in the Department of Electronics and Communication Engineering by Mr. ANNOI ROY (16ETEC004401), Mr. KIRAN PRABHAKAR (16ETEC004028), Mr. NIKHIL C (16ETEC004046) and Mr. M RAHUL NAIDU (16ETEC004032) in partial fulfilment of requirements for the award of B. Tech. Degree in Electronics and Communication Engineering of Ramaiah University of Applied Sciences.*

**July – 2020**

**Mrs. Vasanthavalli S.**

Supervisor

**Dr. S. Malathi**

Head – Dept. of ECE

**Dr. H.M. Rajashekara Swamy**

Dean (I/C)-FET

## Declaration

## All Terrain Semi-Autonomous Rover with Target Acquisition and GPS Tracking

The project work is submitted in partial fulfilment of academic requirements for the award ofB.Tech. Degree in the Electronic and Communication Engineeringof the Faculty of Engineering and Technologyof M. S. Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

| Sl. No. | Reg. No. | Student Name | Signature |
|---|---|---|---|
| 1 | 16ETEC004401 | Annoi Roy | [illegible] |
| 2 | 16ETEC004028 | Kiran Prabhakar | [illegible] |
| 3 | 16ETEC004046 | Nikhil C | [illegible] |
| 4 | 16ETEC004032 | M Rahul Naidu | [illegible] |

**Date:** **30 July 2020**

## Acknowledgements

We would like to express our heartfelt gratitude to everyone who helped us to guide this project to completion. Firstly, a special thank you to our academic supervisor and mentor “Mrs. Vasanthavalli” for carefully guiding us at every step of the way.

We would also like to thank “Dr. H. K. Narahari” for helping us with a major part of the project and without whom we wouldn’t have been able to achieve our goals. Our utmost gratitude to all the teachers in the Electronics and Computer Science department who lent their ears to our endless questions.

We are grateful to our Electronics and Communication Engineering Department Head, “Dr.S. Malathi” and theDean (I/C) of Faculty of Engineering and Technology “Dr.H. M. Rajashekara Swamy” for giving us the opportunity to explore new concepts, build this project and learn immensely in the process.

Lastly, nothing would be possible without the encouragement and invaluable support of our parents, throughout the course of the making of this project.

## Summary

The rover can be interpreted as an extension of any human being's working that can be used to operate under harsh conditions; it can be used for many applications where it may be inconvenient, dangerous, or impossible to have a human operator present. Based on its application, rovers will generally include the following components: platform, sensors, control systems, guidance interface, communication links, and systems integration features.

Now, this rover consists of an aluminum chassis with the mechanical attachments for mobility and a range of sensors for operations and is remotely controlled via interface.

Vision is the most dynamic of all our senses and provides a lot of information about the world around us. The information that an image provides can pave way for further processing and analysis which can be used to automate machines, control computers and find ways to make our daily routines much easier.

The object detection system using Arduino uses sound waves emitted from an ultrasonic sensor to identify the distance and angle of an obstacle that the ultrasonic waves encounter. An alert is provided using a buzzer and LED's to the user as the obstacle is encountered. The image processing system uses the algorithm to identify the colour of the object, draw a contour around the boundary of the object and identify its centre. Once the target is locked, the servo motors follow the object as it moves and hence establish a lock-and-follow mechanism. An input from the keyboard can activate a LASER to fire at the object, making it a LASER turret.

Object detection and real-time tracking is essential in this modern world to track trespassers and intruders of all sorts and ensure security to the users. However, the turret is not completely automated as the final decision-making is in human control. Apart from security purposes, segregation of defective products (based on colour) in several industries can also be subject of concern which can be addressed using this product. These real-time tracking visuals are used to form a live display feed to the user.

Now, finally the GPS telemetry is considered essential because under the said conditions it can effused to pin-point the rover's position even when it is out of visual range. This is because the whole objective of using a rover is to operate in environments where it is not possible for humans to go by themselves or operate.

# Table of Contents

# List of Tables

# List of Figures

# Nomenclature

| | |
|---|---|
| *T* | Time (seconds) |
| *C* | Speed of light(meters/second) |
| *Hz* | Hertz |
| *D* | Distance (meters) |

# Abbreviation and Acronyms

| | |
|---|---|
| LASER | Light Amplification by Stimulated Emission of Radiation |
| DEW | Directed-Energy Weapon |
| LED | Light Emitting Diode |
| RPi | RaspberryPi |
| VNC | Virtual Network Computing |
| PWM | Pulse Width Modulation |
| USB | Universal Serial Bus |
| ICSP | In Circuit Serial Programming |
| DC | Direct Current |
| BLE | Bluetooth Low Energy |
| GPIO | General Purpose Input-Output |
| LAN | Local Area Network |
| CSI | Camera Serial Interface |
| PID | Proportional, Integral, Derivative |
| RGB | Red, Green, Blue |
| HSV | Hue, Saturation, Value |
| UGV | Unmanned Ground Vehicle |

# 1. Introduction

**Preamble:-**

This chapter explains the motivation behind this project, its necessity in the real world and how it stands out from other projects of similar kind.

## 1.1 Introduction

Rovers (UGV systems) could be based upon any of a number of characteristics of each system, including: the purpose of the development effort (often, but not always, the performance of some application-specific mission); the specific reasons for choosing a UGV solution for the application (e.g., hazardous environment, strength or endurance requirements, size limitation); the "long pole" technological challenges, in terms of functionality, performance, or cost, posed by the application; the system's intended operating area (e.g., indoor evironments, anywhere indoors, outdoors on roads, general cross-country terrain, the deep seafloor, etc.); the vehicle's mode of locomotion (e.g., wheels, tracks, or legs); how the vehicle's path is determined (i.e., control and navigation techniques employed).

Image processing has been an evolving field of study over the recent years. In the past five years, there has been a significant increase in the level of interest in image morphology, neural networks, image recognition, data compression and knowledge-based image analysis systems. Image capture and analysis aim to duplicate the effect of human vision by digitally perceiving images. But why would one need these techniques?

To process an image simply means to extract information from it. This information would help one to understand the image and further analyse future images of similar kind. Using multiple images from the same area would increase one’s understanding of the phenomenon going on in the area as well as help to predict what is going to happen in the future in the same area. This knowledge can help scientists train machines to analyse images on their own and take decisions based on the images captured.

A LASER turret is a Directed-Energy Weapon (DEW) that damages its target with highly focussed energy, in this case, a LASER. This DEW can be used discretely, as it travels at speed of light, hence has near infinite range and is only slightly altered by gravity.

In this project, these two ideas are combined along with an additional module for Object detection to create a final Autonomous LASER turret with Object detection and real-time tracking. The object detection module can scan any area 180 degrees and detect an object in the range of 200 centimeters. In addition, servo motors in the image processing module can provide full 180-degree rotation in the left-right as well as up-down directions. The LASER used to point at the centre of the detected object does it with remarkable accuracy.

## 1.2 Necessity:

A number of applications call for machines that can move to a desired area and then perform some sort of work involving manipulation or using any of a variety of tools ("effectors" is the robotic terminology). The issues involved in performing manipulation or other work without a human present often dominate over the UGV navigation and control issues, and tend to be application specific.Application domains addressed by continuing major development efforts include: The nuclear industry, doing work in areas with radiation levels dangerous to human workers. Military heavy equipment for moving dirt under enemy fire, such as repairing craters in a runway or breaching a minefield or other barrier. Strong manipulators for moving and loading heavy items such as ammunition. Explosive ordnance disposal (EOD) -- manipulators capable of dealing with packages containing suspected bombs, unexploded ordnance, etc.

In this modern age, with the technology boom and social media taking over the world, security has become an important matter of concern. Providing a secure environment to safeguard ourselves and our valuables has become essential.

With the advancement in technology, security systems have become automated. People travel halfway across the world trusting their automated machines to keep their precious valuables secure from any sort of threat. This has resulted in machines taking life-and-death decisions on their own and hence reducing the possibility of human error altogether.

Keeping the above requirement in mind, a prototype of a security system is built. This system monitors and tracks a potential threat and can neutralize the threat upon receiving instructions from the user.

## 1.3 Significance:

Autonomous LASER turret with object detection and real-time tracking is a camera-based monitoring system that uses image processing techniques to recognize a moving object within a specified range and when this module is mounted on a rover controlled remotely it packs an extra punch. This prototype stands out amongst other rovers and security systems due to the following reasons-

1) Exact position of the object perceived as a threat is provided to the user in terms of its instantaneous distance and angle from a point of reference.
2) Alerts in the form of LEDs and buzzer noise are provided depending on the distance of the threat from the turret.
3) A threat of a particular colour (blue, in this case) is recognized and its movement is tracked. This can be used in specific applications where a colour sensitive trespasser is of concern.
4) A real-time video of the threat is displayed on the screen of the user so as to provide full information about the intruder. The final decision of firing of the turret is left to the user so as to avoid any false alarms.
5) The range of operations increase as it is controlled by tele-operation.

# 2.Background Theory

**Preamble:-**

This chapter focuses on the research done before commencing this project, including background theory of all the main components needed for it. Also, the working of Arduino and RaspberryPi and the algorithms of the codes that are dumped in them have been looked into in this chapter.

## 2.1 Background survey of components

### *2.1.1) Ultrasonic sensor*

An ultrasonic sensor is an instrument that measures the distance of an object using ultrasonic sound waves. It uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High frequency sound waves reflect from boundaries of the object to produce distinct echo patterns.

<u>Working principle</u>:-

Ultrasonic sensors work at a frequency above the range of human hearing. The sensor determines the distance of the target by measuring time lapses between sent and received ultrasonic pulses. The distance is calculated by-

$$D = \frac{1}{2} T \times C$$

Where, D = Distance,

T = Time

C=Speed of Light

At 20°C, the speed of sound is 343 meters/second, but this varies depending on temperature and humidity and hence a general consideration is made.

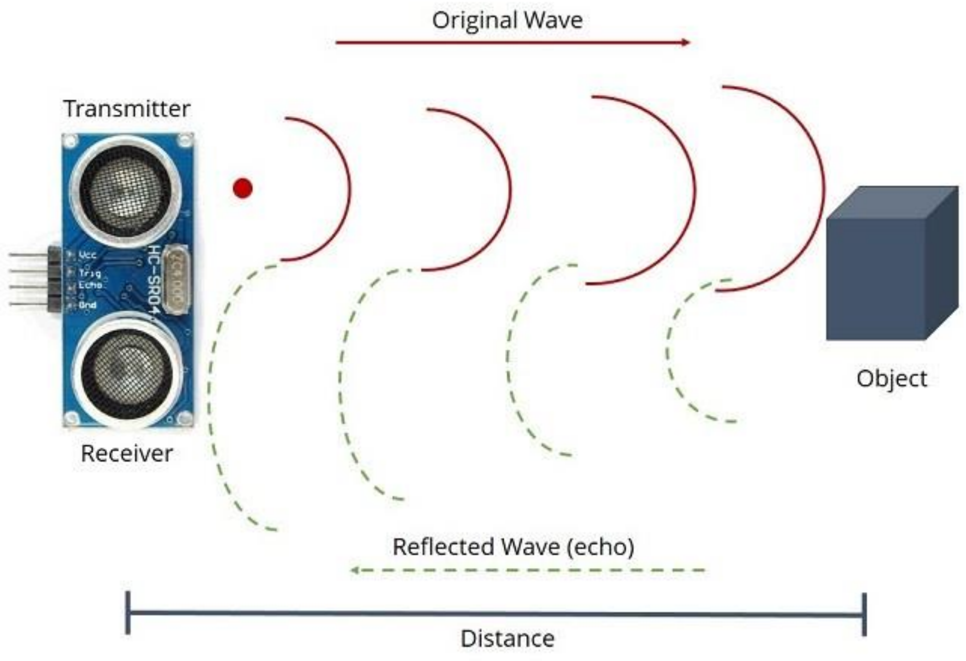


**Figure 2.1: Working of an ultrasonic sensor**

Ultrasonic sensors are independent of light, smoke, dust, colour and material (as long as it doesn’t absorb sound waves) and it is considered superior to infrasonic sensors in all respects.

### *2.1.2) Servo motor*

A servo motor is a rotary actuator that gives precise control in terms of angular position, acceleration and velocity. It uses the mechanism of a regular motor with the help of a sensor for position feedback.Servo motors are small, energy efficient and can be controlled easily using a code of our choice.

<u>Features</u>: -

The servo circuitry is built right inside the motor unit and has a positionable shaft, which usually is fitted with a gear. The motor is controlled with an electric signal which determines the amount of movement of the shaft. Inside there is a pretty simple set-up: a small DC motor, potentiometer, and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes. So the control circuit can precisely regulate the extent of movement and its direction. The motor's speed is directly proportional to the difference between the actual position and the desired position. Farther the desired position, the faster it'll rotate, making it efficient.

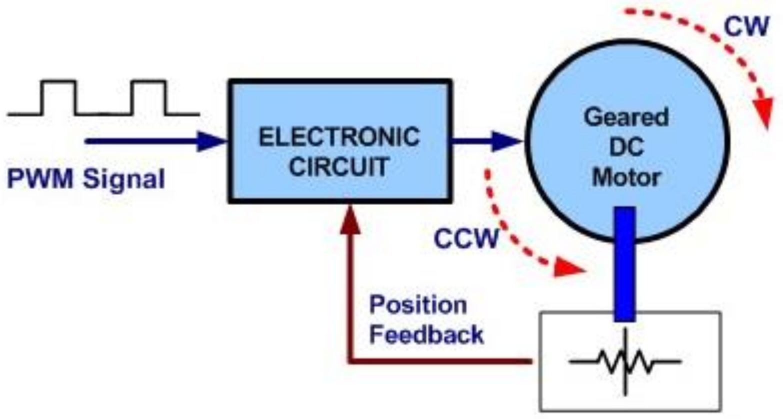


**Figure 2.2: Servo motor block diagram**

<u>Servo control</u>: -

A servo motor has three wires coming out of it, for connection. The brown and red wires should be connected to +-5V DC supply whereas the orange wire should be given to the signal (in this case PWM) for control.

Servos are controlled by sending an electrical pulse of variable width (pulse width modulation PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the motor determines position of the shaft, and based on the duration of the pulse sent via the control wire, the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

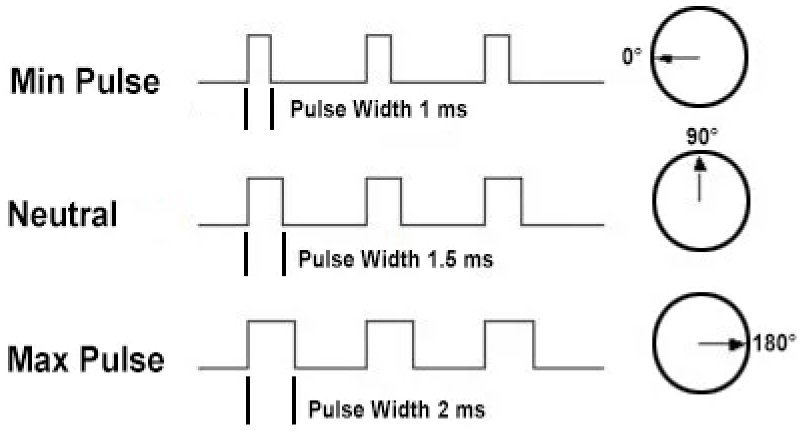


**Figure 2.3: Servo control mechanism**

### *2.1.3) DC Geared Motor*

Geared DC motors can be defined as an extension of DC motor which is a type of electrical machinery that is able to produce high torque at low speed motor output. A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction.

**Figure 2.4: DC Geared motor**

A motors performance and gearbox performance are combined into one graph by displaying three specific parameters. These three parameters are speed, torque and efficiency. These performance curves are essential when selecting a gear motor for your application.

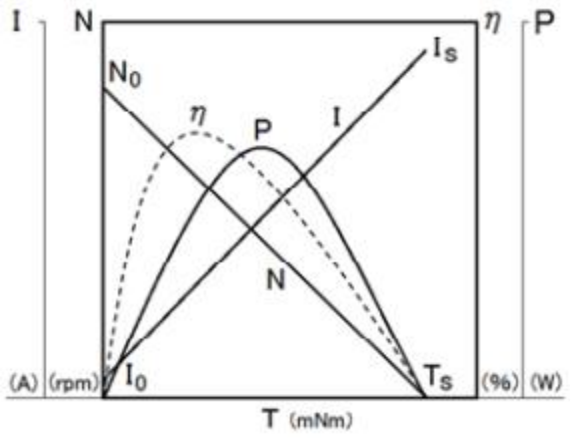


**Figure 2.5: Performance Curve of DC geared motor**

*Speed/Revolutions* (N) – (*unit: rpm)* indicated as a straight line that shows the relationship between the gear motors torque and speed. This line will shift laterally depending on voltage increase or decrease.

*Current* (I) – (*unit: A)* indicated by a straight line, from no load to full motor lock. This shows the relationship between amperage and torque.

*Torque* (T) – (*unit: gf-cm)* this is the load borne by the motor shaft, represented on the X-axis.

*Efficiency* (η) – (*unit: %)* is calculated by the input and output values, represented by the dashed line. To maximize the gear motors potential it should be used near its peak efficiency.

*Output* (P) – *(unit: W)* is the amount of mechanical energy the gear motor puts out.

### *2.1.4) Battery*

The Battery is the main power source to the entire rover. A battery of high capacity is required to operate the rover alongside the weight of the battery cannot be high. Considering these parameters  Li-Po (Lithium Polymer) batteries are used. They have an output current capacity of current capacity of 5200mAh.

## 2.2) Open source boards:

### *2.2.1) Arduino Uno*

Arduino Uno is a single-board microcontroller based on ATmega328 microcontroller chip. It is equipped with 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button. It has a wide range of applications in fields like Robotics to Home and industry Automation.

The Arduino is connected to the computer and code is uploaded. Now every time power supply is given to the Arduino, code is automatically executed without any physical connection to a computer.

### *2.2.2) RaspberryPi*

RaspberryPi is a 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, fast Ethernet and Power-over-Ethernet support. It is a small single-board computer which runs on Linux having a set of GPIO (general purpose input output) pins that allows the user to control electronic components for physical computing. Being the world’s least expensive and most versatile computer, RPi can run 24/7 without a cooling fan and can be fast enough to surf web pages.

RaspberryPi 3B+ model is used in our project. VNC viewer was installed in the laptop to access the RPi terminal and visualize the processes going on in the Pi. First, a LAN connection to the router was used to access the terminal and upload the code. Then, wireless hotspot connection was established so that code could be run without any physical connection to the Pi. Once the power supply is given to the RPi and wireless hotspot connection is established, the RPi terminal can be accessed using VNC viewer and the code is run without any hassles.

### 2.2.3)Pixhawk PX4

Rovers (UGV) are generally considered Remote-Operated and Autonomous, although Supervisory Control is also used to refer to situations where there is a combination of decision making from internal UGV systems and the remote human operator.

The main controller of the rover is Pixhawk PX4 Flight controller. The Pixhawk PX4 has its own designated software called Mission Planner. Mission Planner is a ground control station for Plane, Copter and Rover. It is compatible with Windows only. Mission Planner is used as a configuration utility for the rover. The mission Planner is an Open Source Software used to calibrate the Flight Controller with the various components like the motors and its controllers. The Mission Planner software is also used to calibrate the rover with the GPS module and the Telemetry module.

Specifications

- **Processor**
  - 32-bit ARM Cortex M4 core with FPU
  - 168 Mhz/256 KB RAM/2 MB Flash
  - 32-bit failsafe co-processor

  **Sensors**
  - MPU6000 as main accel and gyro
  - ST Micro 16-bit gyroscope
  - ST Micro 14-bit accelerometer/compass (magnetometer)
  - MEAS barometer

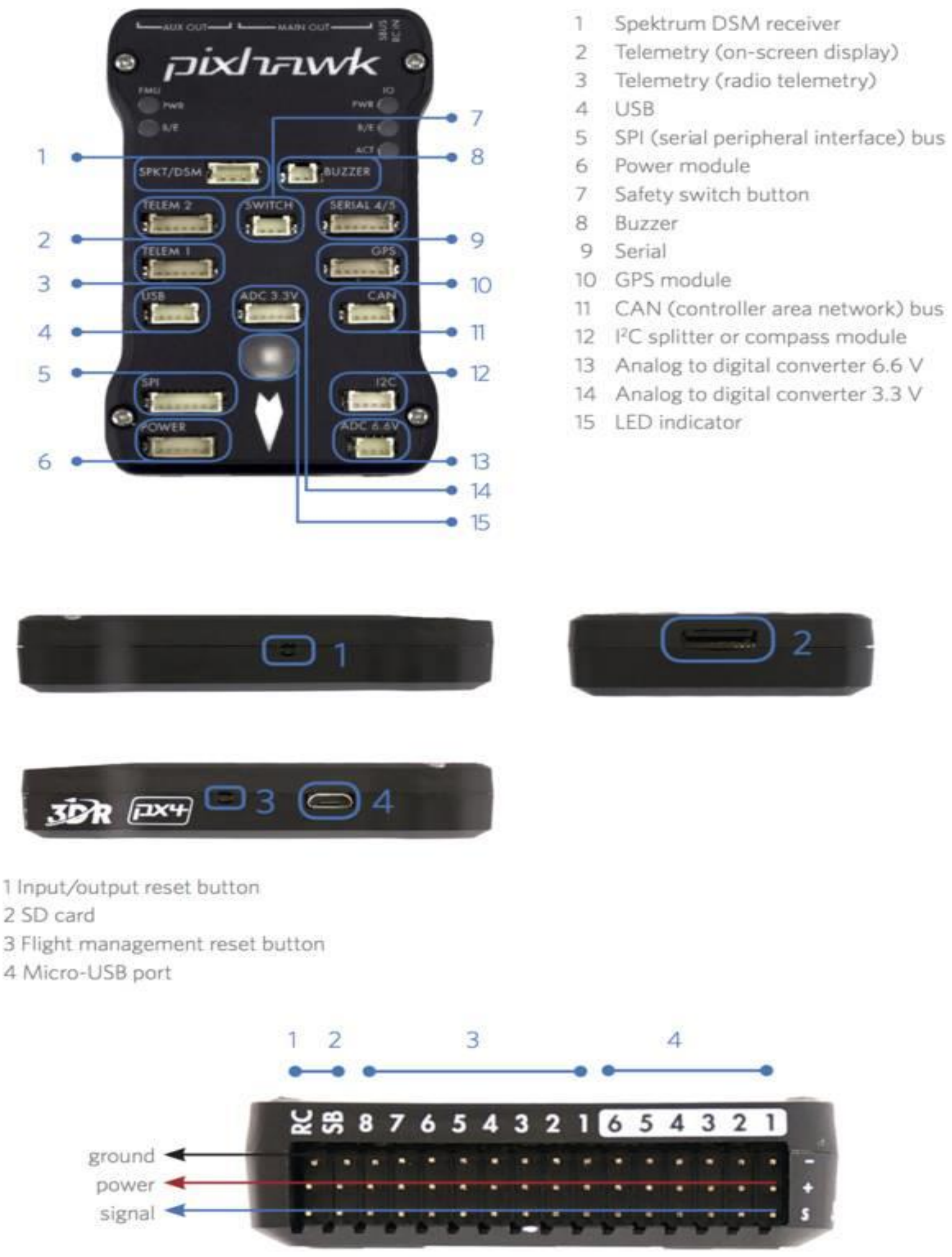
  **Power**
  - Ideal diode controller with automatic failover
  - Servo rail high-power (7 V) and high-current ready
  - All peripheral outputs over-current protected, all inputs ESD protected

  **Interfaces**

- 5x UART serial ports, 1 high-power capable, 2 with HW flow control
- Spektrum DSM/DSM2/DSM-X Satellite input
- Futaba S.BUS input (output not yet implemented)
- PPM sum signal
- RSSI (PWM or voltage) input
- I2C, SPI, 2x CAN, USB
- 3.3V and 6.6V ADC inputs

**Dimensions**

- Weight 38 g (1.3 oz)
- Width 50 mm (2.0”)
- Height 15.5 mm (.6”)
- Length 81.5 mm (3.2”)



1 Spektrum DSM receiver
2 Telemetry (on-screen display)
3 Telemetry (radio telemetry)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator

1 Input/output reset button
2 SD card
3 Flight management reset button
4 Micro-USB port

1 Radio control receiver input
2 S.Bus output
3 Main outputs
4 Auxiliary outputs

**Figure 2.6: View and brief description of Pixhawk PX-4 controller**

# 3. Aim and Objectives

- **Title**
  - All Terrain Semi-Autonomous Rover with Target Acquisition and GPS Tracking
- **Aim**
  - To develop an all-terrain semi-autonomous robotic vehicle with real time video feed and computerized target acquisition and GPS tracking.
- **Objectives**
  - To conduct a literature survey of a rover module and various add-ons.
  - To design the rover module with all the attachments.
  - To design the prototype & to develop the code for all the modules in the project.
  - To develop a real time video feed and integrate the modules with the rover.
  - To test and evaluate for sensitivity and accuracy of the developed integrated model and validate its performance in real-time scenarios.
  - To conduct a literature survey to understand the functioning of Arduino and RaspberryPi and concepts of image processing.
  - To develop a design for object detection and image processing modules.
  - To detect the presence of an object in the specified range and display the distance and angle from a point of reference.
  - To analyze the detected object, target lock using image processing techniques, draw a contour, identify its center and point a LASER at it.
- **Methods and Methodology/Approach to attain each objective**

  ****(PTO)**

**Table 3.1 Method/Methodology used to accomplish the objectives.**

| Objective No. | Statement of the Objective | Method/ Methodology | Resources Utilised |
|---|---|---|---|
| 1 | To develop a design for object detection module | • Selection of Arduino and Ultrasonic sensor for the proposed application.<br>• Assembling the components to build the module<br>• Connections made between Arduino and all other components. | • Arduino programming tool<br>• LCD Screen<br>• LEDs and Buzzer |
| 2 | To develop a design for image processing modules | • Selection of Raspberry Pi and Servo motor for the proposed application.<br>• Interfacing RaspberryPi board with VNC server.<br>• Assembly of the servo motors to build a pan tilt kit. | • Raspbian Operating System<br>• Open CV library<br>• RaspberryPi camera module<br>• LASER |
| 3 | To detect the presence of an object in the specified range | • Selection of suitable algorithm for object detection<br>• Developing a code for implementation in microcontroller<br>• Testing for 180◦ movement of the motor and working of the sensor | • Arduino programming tool<br>• Object for detection |

| | | | |
|---|---|---|---|
| 4 | To display the distance and angle of the object from a point of reference | • Selection of suitable algorithm and developing a code for proposed objective<br>• Interfacing LCD Display with Arduino<br>• Testing for the suitable indication to the user | • Arduino programming tool<br>• LEDs and Buzzer |
| 5 | To analyze the detected object, target lock using image processing techniques | • Selection of suitable algorithmand developing a code for image processing<br>• Interfacing camera module and the LASER with the RaspberryPi<br>• Testing for the Two-Degree motion of the servo pan tilt kit | • Open CV library<br>• VNC Viewer software<br>• Blue object for detection |
| 6 | To draw a contour, identify its center and point a laser at the object | • Developing a code for implementation of object recognition<br>• Testing for the real-time display of the detected blue object<br>• Testing for the LASER operation | • VNC Viewer software<br>• Blue object for detection<br>• Computer screen for viewing |

# 4. Design and Development

**Preamble:-**

The system is split into three distinct modules:

❖ Target Acquisition module; it is further divided into 2 separate modules.

- Object Detection
- Image processing

❖ GPS Telemetry

❖ Surveillance Module

Each of their design and functioning is explained elaborately in this chapter.

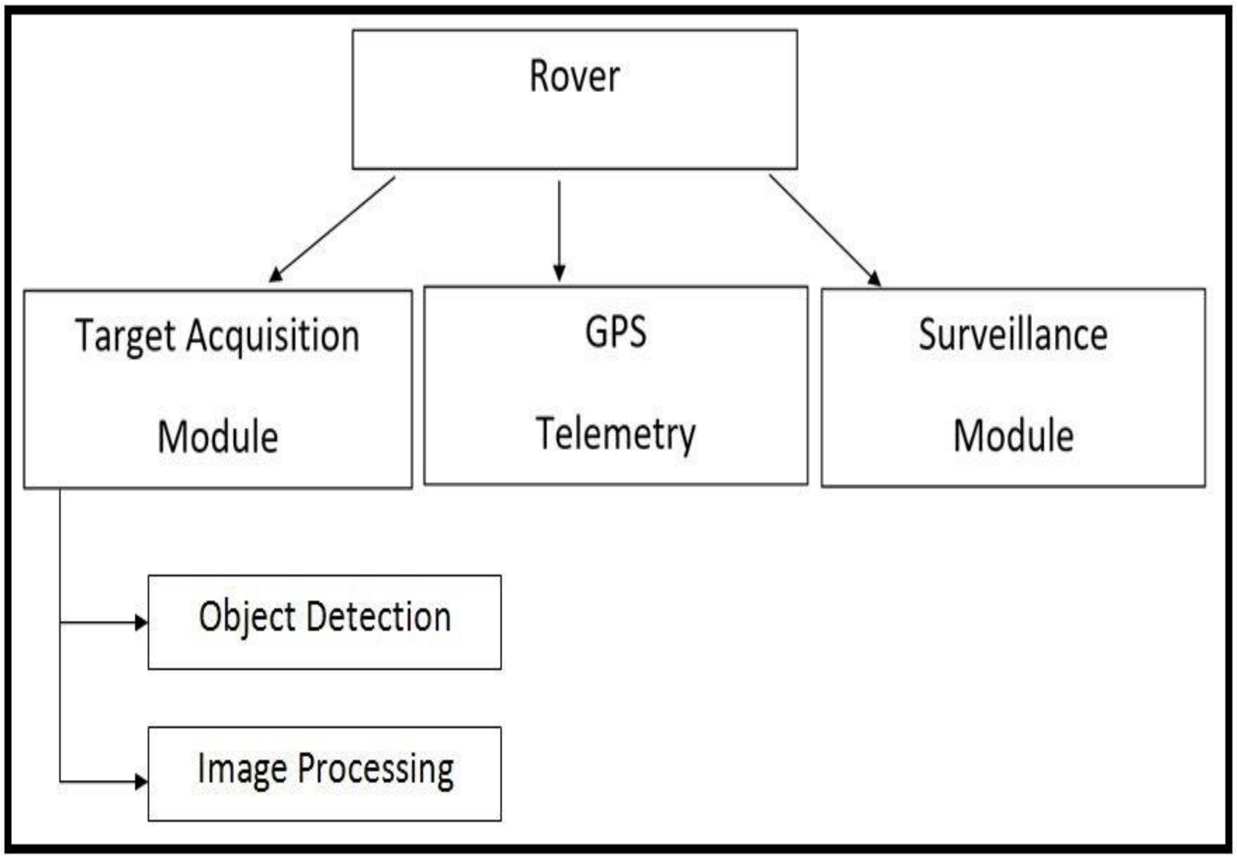


**Figure 4.1: Block Diagram of Project model.**

## 4.1: Design of the Rover

In this section the proposed design and construction of the rover’s body is illustrated. The design of the rover is inspired by the ICV BMP II or the Sarath tank.

The design of the rover is tank based solely so that it is compatible with the various environmental terrains to make it manoeuvrable on different landforms.

The Following is the dimensional analysis of the rover.

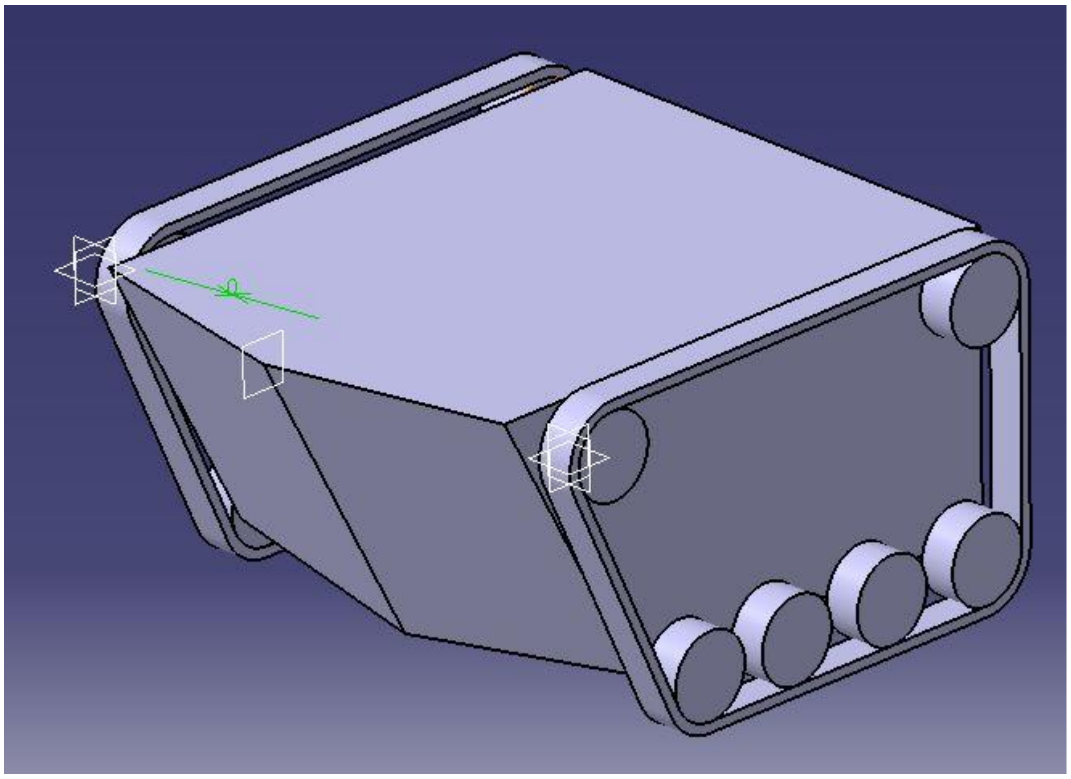


**Figure 4.2: Design of the Model.**

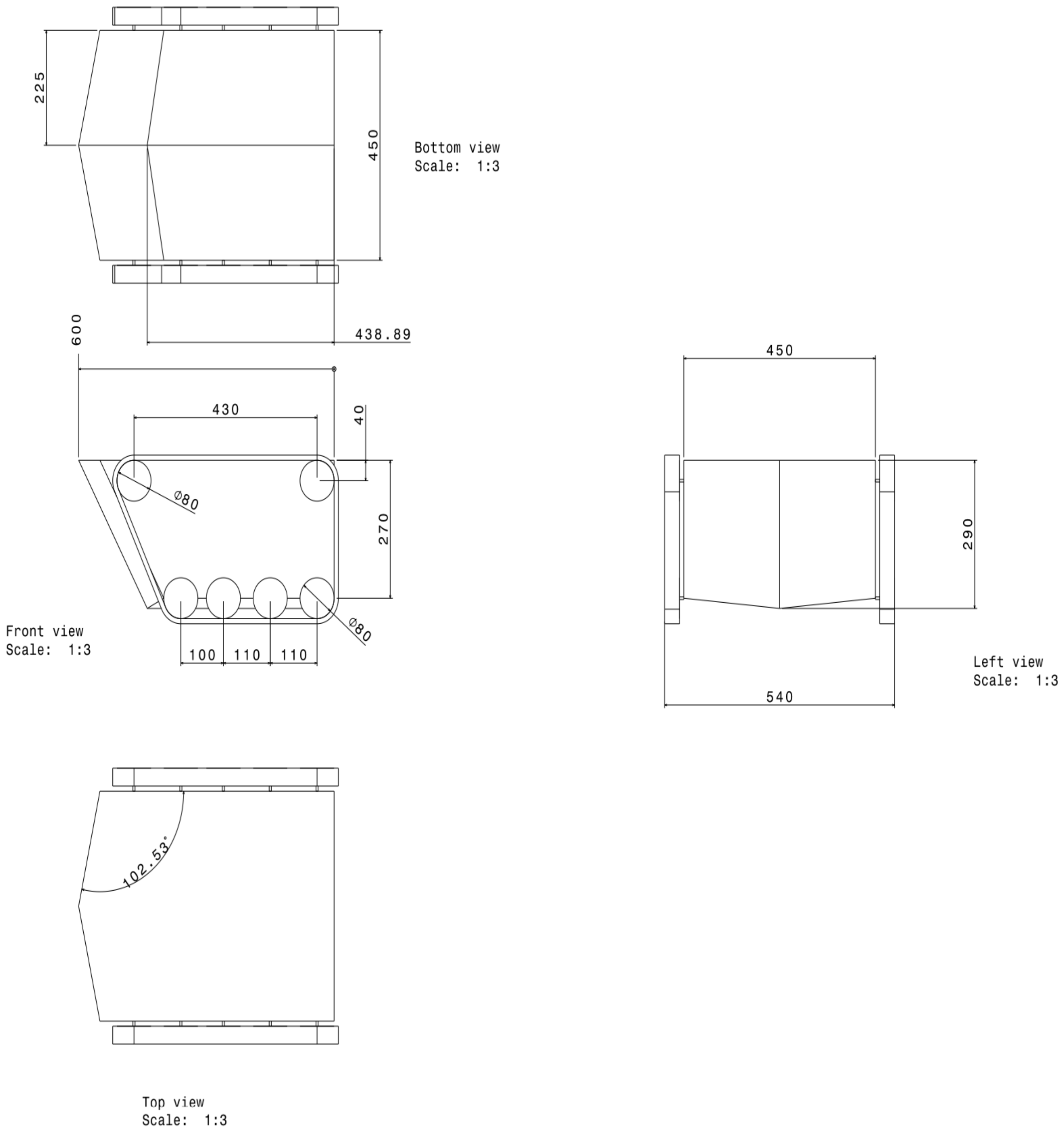
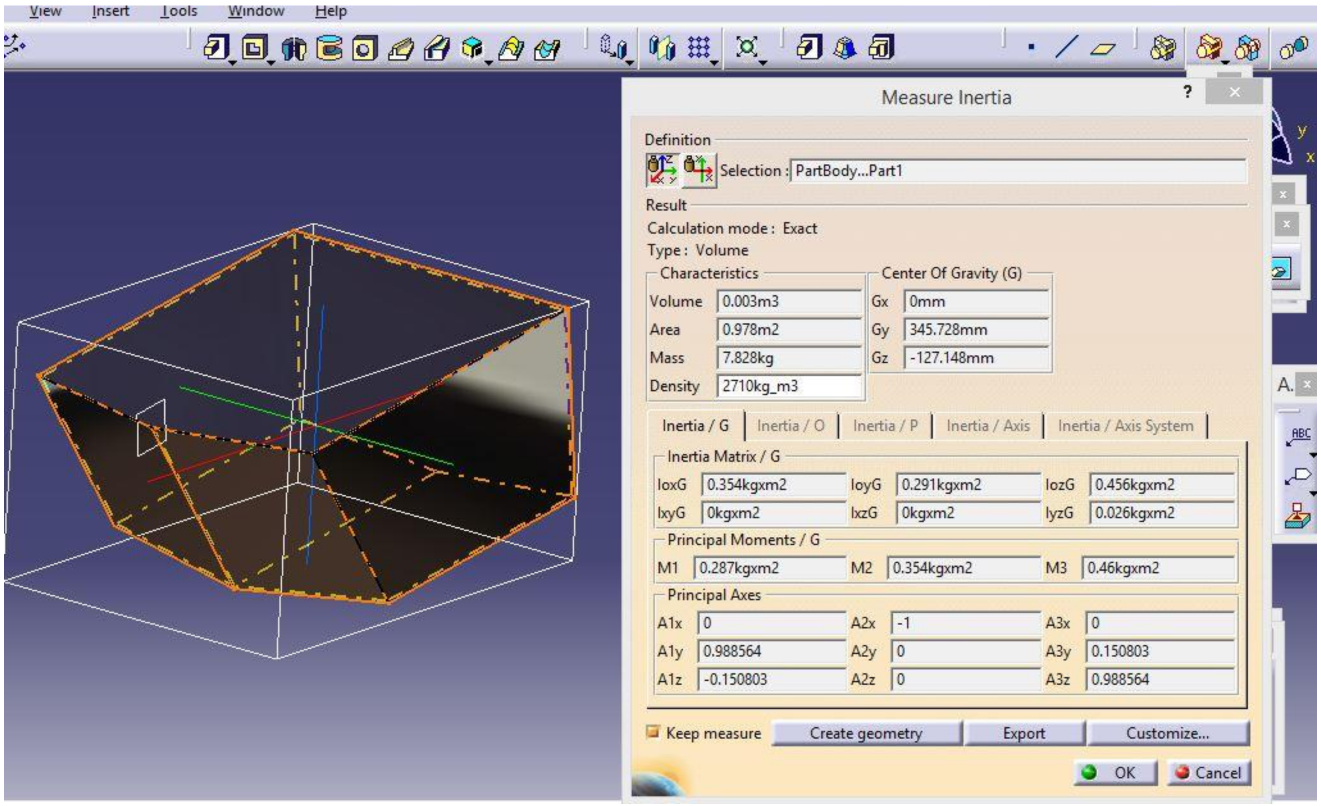


**Figure 4.3: The dimensions of the model as designed in AutoCAD.**

The Mechanical Parameters of the hull(body) of the rover is as follows-



Measure Inertia

Definition

Selection : PartBody...Part1

Result

Calculation mode : Exact

Type : Volume

| Characteristics | | Center Of Gravity (G) | |
|---|---|---|---|
| Volume | 0.003m3 | Gx | 0mm |
| Area | 0.978m2 | Gy | 345.728mm |
| Mass | 7.828kg | Gz | -127.148mm |
| Density | 2710kg_m3 | | |

Inertia / G | Inertia / O | Inertia / P | Inertia / Axis | Inertia / Axis System

Inertia Matrix / G

| | | | | | |
|---|---|---|---|---|---|
| IoxG | 0.354kgxm2 | IoyG | 0.291kgxm2 | IozG | 0.456kgxm2 |
| IxyG | 0kgxm2 | IxzG | 0kgxm2 | IyzG | 0.026kgxm2 |

Principal Moments / G

| | | | | | |
|---|---|---|---|---|---|
| M1 | 0.287kgxm2 | M2 | 0.354kgxm2 | M3 | 0.46kgxm2 |

Principal Axes

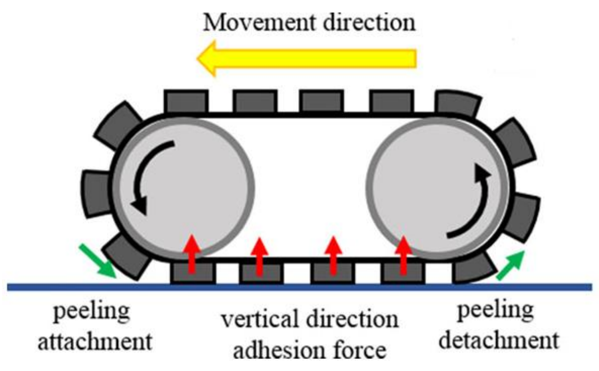
| | | | | | |
|---|---|---|---|---|---|
| A1x | 0 | A2x | -1 | A3x | 0 |
| A1y | 0.988564 | A2y | 0 | A3y | 0.150803 |
| A1z | -0.150803 | A2z | 0 | A3z | 0.988564 |

**Figure 4.4: Mechanical parameters of the hull of the rover.**

The entire body of the rover is made of Aluminium for its light weight and durability against any impulse.

The complete hull of the rover is made hollow so that it can float on the water and can be controlled even on any water surface.

The rover is driven by caterpillar tracks, as Tracked vehicles can move easily over rough terrain because the track makes contact with a wide area of the ground.It also facilitates ease of movement on different kinds of terrain.Caterpillar tracks work on the same principle as a conveyer belt. The power derived from the battery is used to run the motor which rotates one or more steel sprockets, which move a track made up of multiple links.



**(a)** **(b)**

**Figure 4.5: (a) Working principle of caterpillar tracks;**

**(b) Rubber caterpillar tracks proposed in this design.**

### 4.1.2: Motors and ESC

To be able to easily move the rover which weighs around 7-8kg, a motor with high power is required in this case.

Since the rover is skid steering based the ideal configuration of the motor would be low speed and high torque.

The above configuration is such that it facilitates enough track grip over the surface on which the rover is moving upon.

Hence the ideal class of motor selected is a DC Geared Motor which is capable of 300 rotations per minute working on 12V DC or 3S Li-po Battery.

The motor is linked to the flight controller via a Motor Controller or an Electronic speed Controllers which maintains the constant speed of the motor and controls the speed of the motor accordingly with the receiving instructions.

### 4.1.3: Software

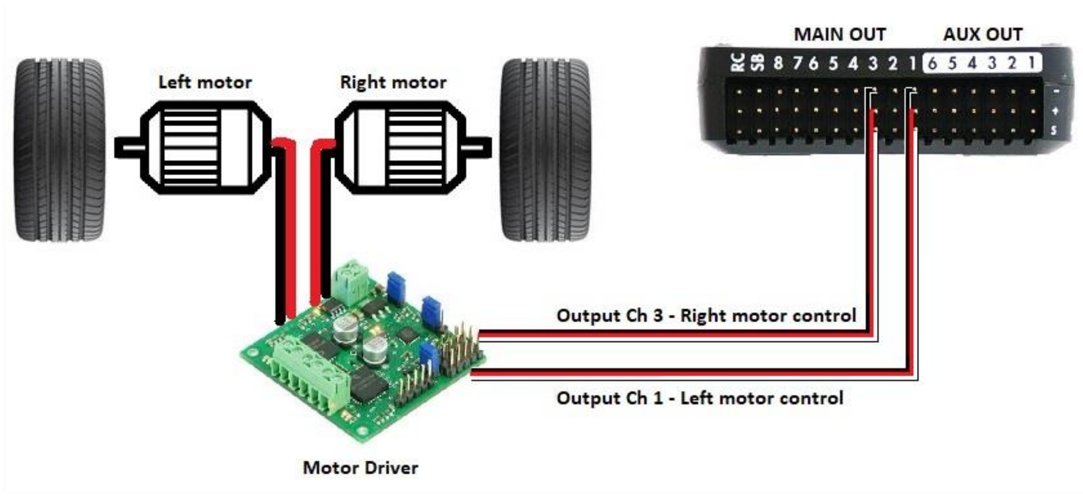
The mission Planner is the Open Source Software used to calibrate the Flight Controller (PixHawk PX-4) with the various components like the motors and its controllers. Mission Planner is a ground control station for Plane, Copter and Rover. Mission Planner is used as a configuration utility for the rover. The Mission Planner software is also used to calibrate the rover with the GPS module and the Telemetry module. The various waypoints on which the rover moves in auto mode are setup on the mission planner software.

### 4.1.4: Connection of the Motors.

The rover is identical to a tank structurally speaking;hence parallel driven motor setup is used which need to be powered individually.

**Skid Steering**

"Skid steering" vehicles control their direction and forward/reverse motions by varying the speed of two (or more) independent motors. For these style rovers the left wheel should be connected to RC Output 1 and the right wheel should be connected to RC Output 3.



**4.6: Illustration of the motor connections.**

For this setup these parameters values will need to be set.

- SERVO1_FUNCTION = 73 (Throttle Left)
- SERVO3_FUNCTION = 74 (Throttle Right)

## 4.2 Object detection module:

### 4.2.1 Design:

This module primarily consists of an ultrasonic sensor, a servo motor, LCD display, two LEDs, a buzzer and an Arduino board.

To detect the object from a height and make the model more presentable, a tower made of a cardboard roll is built. It is fixed onto the metal plate base on which the entire model is set up. The servo motor is suitably fixed on top of the tower allowing a full 180 degree rotation without any hindrance. A small cardboard plank holding the ultrasonic sensor at its tip is placed on top of the servo motor.

The LCD display, Arduino board and the breadboard with a buzzer and two LEDs are fixed on the metal plate base and connections, as shown in figure 4.1 are made using jumper wires.

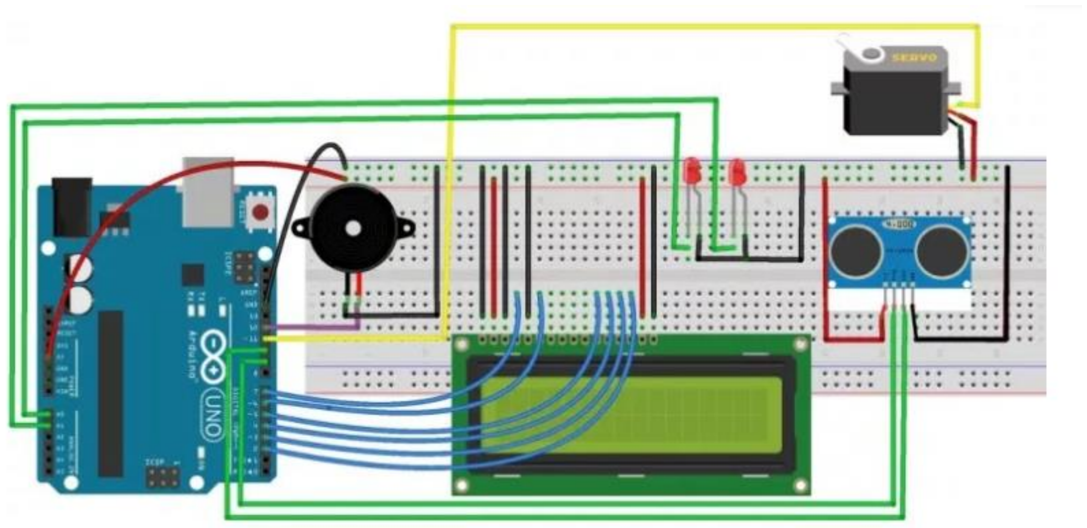


**Figure 4.7: Circuit diagram of Object Detection module**

### 4.2.2) Code description and implementation:

The Arduino code begins with the definition of functions for servo motor, LCD display and initializing the input and output pins used in the code.

```
Servo myservo;
LiquidCrystallcd(7, 6, 5, 4, 3, 2); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
```

```
const int trigPin = 9;
const int echoPin = 10;
const int motor = 11;
const int buzzer = 12;
const int ledPin1 = 14;
const int ledPin2 = 15;
float distanceCm, DistanceSec,duration;

void setup() {
myservo.attach(motor); // Attach motor to pin 11
lcd.begin(16,2); // Initialises interface between lcd and Arduino pins
//Defining input and output ports
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(buzzer, OUTPUT);
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
```

Once the power supply is given to the Arduino, it initiates movement of servo motor to the 0-degree position and begin its rotation towards 180 degrees in small increments of 1 degree. It also sends a HIGH pulse of a 10µs width to the Trigger pin of the ultrasonic sensor making it generate a series of ultrasonic waves which propagate though the air until an obstacle is encountered. The Echo pin of the sensor remains at HIGH position during transmission, thereby measuring the round trip timing of ultrasound which helps in determination of distance of the object. The ultrasonic waves deflect from the obstacle and return in the opposite direction towards the Echo pin making it LOW once the waves come back.

The for loop in the code that is responsible to carry out the above actions is as mentioned below-

```
int pos = 0;
void loop() {
for (pos = 0; pos<= 180; pos += 1) { // Allows rotation from 0 to 180 degrees in
steps of 1 degree
myservo.write(pos); // Recording the position of servo motor
digitalWrite(trigPin, LOW); //Sends pulse every 2ms if object not detected
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); //Sends pulse every 10ms if object detected
delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH); //Time taken for pulse to come back to echo pin
recorded
distanceCm= duration*0.034/2;
```

The distance of the obstacle is noted and compared with the range specified in the code. If the distance of the object is within the range, the rotation of the servo is stopped for 10 seconds. If the object distance is less than half the range, the buzzer beeps for 700ms, the second LED turns on and the distance and angle of the object measured is displayed on the LCD screens with appropriate suffixes.This is executed using the following code-

```
DistanceSec=20;
if (distanceCm<= DistanceSec)
{
if(distanceCm<= DistanceSec/2)
{
tone(buzzer, 10); // Send 1KHz sound signal
digitalWrite(ledPin1, LOW); //LED1 off
digitalWrite(ledPin2, HIGH); //LED2 on
delay(700);
noTone(buzzer); // Stop sound
lcd.setCursor(0,0); // Set cursor position as (0,0)
lcd.print("Distance: "); // Print "Distance: " on LCD
lcd.print(distanceCm);
lcd.print(" cm "); // Add suffix "cm" after diplaying distance
delay(10);
lcd.setCursor(0,1);
lcd.print("Angle : "); // Print "Angle: " on LCD
lcd.print(pos);
lcd.print(" deg "); //Add suffix "deg" after displaying position
delay(2000);
```

A similar code is written for when then object distance is greater than half of the range but still within the range, where the buzzer beeps for lesser time and the first LED glows. If the object is out of the range then the servo motor does not stop and neither the buzzer beeps nor the LEDs glow. This is done by the given below code –

```
digitalWrite(buzzer, HIGH);
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin1, HIGH);
delay(100);
digitalWrite(buzzer, LOW);
lcd.setCursor(0,0);
lcd.print("Distance: ");
lcd.print(distanceCm);
lcd.print(" cm ");
delay(10);
lcd.setCursor(0,1);
lcd.print("Angle : ");
lcd.print(pos);
lcd.print(" deg ");
delay(2000);
```

A flow chart of the working of this module is given below-

****PTO**

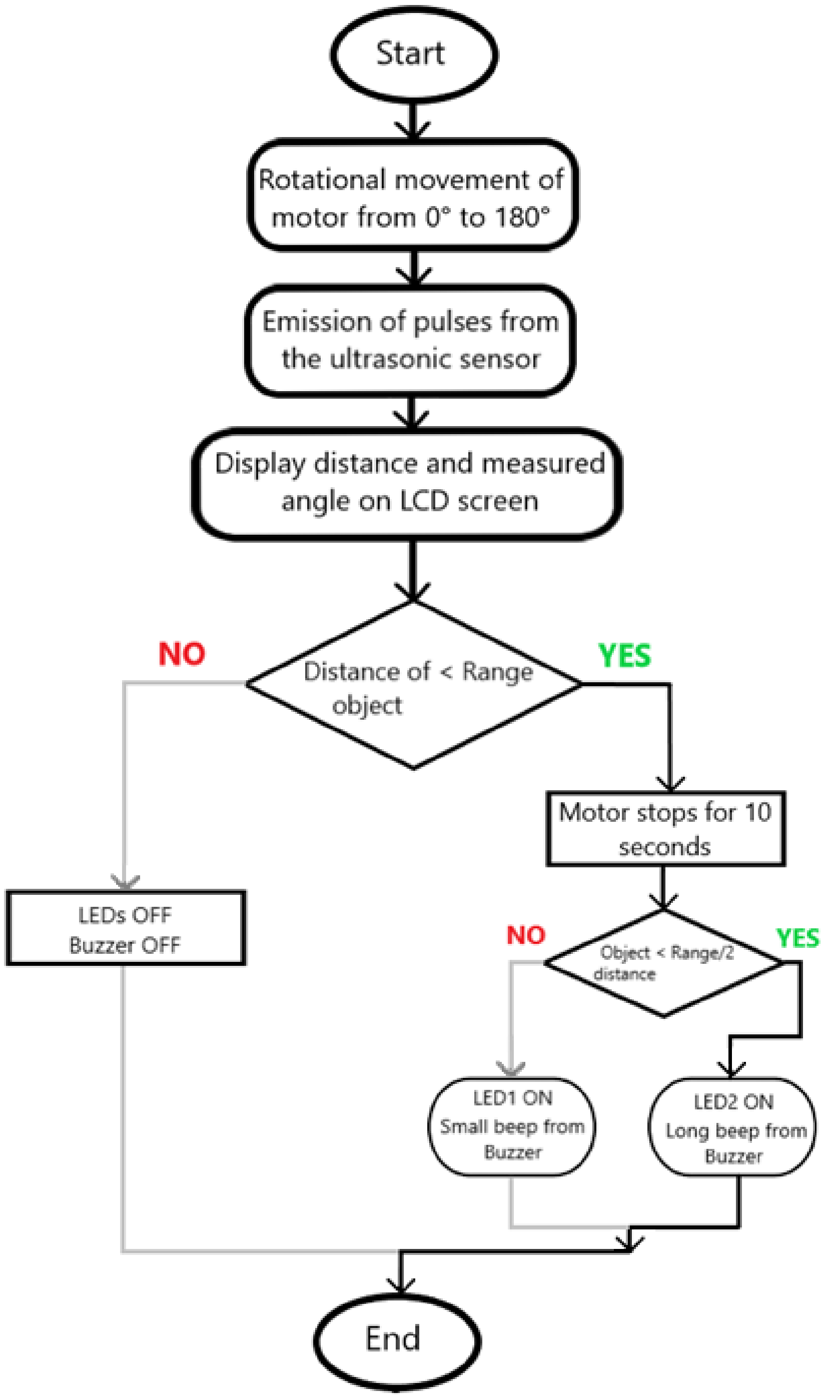

Start
Rotational movement of motor from 0° to 180°
Emission of pulses from the ultrasonic sensor
Display distance and measured angle on LCD screen
NO
Distance of object < Range
YES
Motor stops for 10 seconds
LEDs OFF Buzzer OFF
NO
Object distance < Range/2
YES
LED1 ON Small beep from Buzzer
LED2 ON Long beep from Buzzer
End

**Figure 4.8: Flow Chart for Object Detection module**

### 4.2.3)Working:

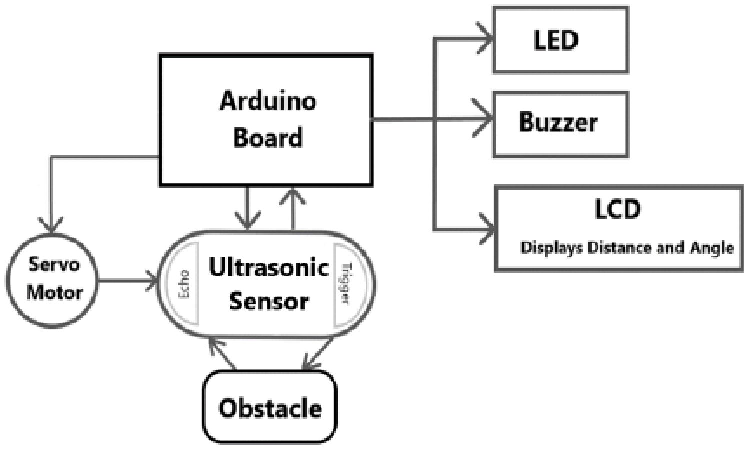


**Figure 4.9: Block Diagram of Object Detection module**

The figure 4.3 shows the working of the object detection module. Its primary objective is to detect the presence of any object that enters its range and alert the user about it.

The Arduino is first fed with the object detection code. On powering up the Arduino and the servo, instruction is given to the motor to start its rotation from the 0 degree position. As the rotation continues from 0 to 180 degrees, the ultrasonic sensor continuously transmits sound waves and receives them back, calculates the distance and displays it instantaneously along with the angle on the LCD screen.

If an object enters the range set in the code, the servo motor halts for 10 seconds and an alert is given to the user depending on the distance at which the object is form the turret. If the object is within the range but in the outer half, the first LED glows and the buzzer beeps for a short while. But the object being within the inner half of the range, the second LED glows and the buzzer beeps for a longer time. Along with this, the exact position of the object (distance and angle) is still being displayed on the LCD screen.

Therefore, the user gets fully informed about the whereabouts of the object and can take further actions depending on that.

### 4.3 Image processing module:

#### 4.3.1) Hardware design

The image processing module consists of a servo pan tilt kit, a RaspberryPi camera module, a laser with a control line and a RaspberryPi.

The servo pan tilt kit consists of two micro servo motors (with feedback) that can each move 180 degrees, plastic housing and some screws. Once put together, the pan-tilt kit was perfect to get the desired motion for the turret; one servo for left-right rotation and the other for up-down tilt. The RaspberryPi camera module is directly attached to the front portion of the tilt kit such that the rotation and tilt of the servos move the camera to the right position. A15-pin ribbon cable from the camera is connected to the Camera Serial Interface (CSI) port of the RaspberryPi, which is housed in a cardboard case beneath the pan tilt kit. Finally, a laser with a control line is mounted to the side of the tilt kit, making it easy to be controlled by software and making our turret, a turret.

The power supply for our servo motors as well as the laser is given from a simple USB cable connected to a laptop. The brown wires from the servos and laser are given to ground and the red wires are given to +5V supply. The orange wires and control line of the laser are given to respective GPIO pins of the RaspberryPi as per the code.

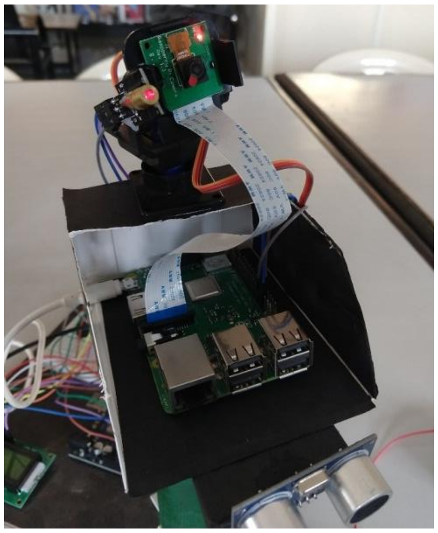


**Figure 4.10: Image processing module**

#### 4.3.2) Software design:

The first part of the image processing code involves importing the necessary libraries needed for the working of this turret. The 'RPi.GPIO' library for GPIO (General purpose input output) pin functioning, 'time' library for time control, 'numpy' library for mathematical operations, 'pigpio' library for generation of hardware PWM signals and so on. The full OpenCV library was installed prior to this in the RPi.

```
Frommultiprocessingimport Process, Queue, Value, Lock, Array
ImportRPi.GPIOasGPIO
Importtime
Importsys
Importnumpyasnp
Importpigpio
Importcv2
Fromdatetimeimport datetime
```

The GPIO pins for the laser as well as the two servos are set up. The laser pin is assigned as GPIO6 and the rotating and tilt servos are assigned GPIO12 and GPIO13 respectively. Hardware PWM signals at 50Hz frequency are given to the servo GPIO pins.

```
GPIO.setmode(GPIO.BCM)
laserPin=6
GPIO.setup(laserPin, GPIO.OUT)
GPIO.output(laserPin,GPIO.LOW)
motorPinR=12
motorPinT=13
pi_hw=pigpio.pi()
pi_hw.hardware_PWM(motorPinR, 50, 80000)
pi_hw.hardware_PWM(motorPinT, 50, 75000)
```

Three functions, 'laser', 'rotate5' and 'tilt5' are defined and their working is coded. They toggle the laser and move the servos in the left-right and up-down directions. The detailed code for this is in the appendix.

The main software design of our turret is can be divided to three main parts: Object recognition, PID control and multi-processing algorithms. These algorithms are explained in depth in the following section.

a) <u>Object recognition algorithm:</u>

The object recognition algorithm detects the largest blue object viewed on camera and extracts its centre position. First OpenCV library is installed, which contains all the processesneeded for this project. The object recognition code can be broken down into the following steps-

1. cv2.VideoCapture() is used to create a video capture object that would return a 640x480x3 picture array whenever videoCap.read() is called. 640x480 is the default image resolution used here with a depth of 3 for three at RGB(red, green, blue) colour space.
2. cv2.cvtColor() is used to convert the picture array from RGB colour space to HSV (hue, saturation, value) colour space because it is easier to detect a colour at HSV than RGB colour space.
3. cv2.inRange() is used to check pixels in the picture array and creates a mask for those in the blue range that is specified in the code. It does so by checking each individual pixel and comparing it with the range. Before the mask is fed into the following time intensive steps, it is checked if the sum of values in the mask is greater than a particular threshold. If it is, it is treated as a valid blue object but if not, all of the following steps are skipped. In this way, not every separate blue pixel is detected and unnecessary computation is avoided.
4. cv2.morphologyEx is called twice to apply the open and close operation on the mask, which is useful to get rid of a lot of background noise, separate connecting objects and solidify an object with its hazy boundaries.
5. cv2.findContours is used to extract the boundary of the objects.
6. Once the boundary of an object is defined, cv2.moments is used to extract the spatial moments that can be used to compute the centre position of an object. Since onlyone object is tracked at a given time, all the contours obtained are looped from cv2.findContours to find the object with the maximum contour which is only passed to cv2.moments for centre extraction.

b) PID control algorithm:

Once the centre position of the object is extracted, the target lock is established. The turret's current position is the centre of the camera (320,240) which is half the image resolution 640x480. As the object moves, knowing the current position and the desired position, the difference can be computed and used as feedback to the servos, thereby achieving tracking. For instance, if the object moves to the top right corner relative to its current position, the servos move incrementally to the right and then to the top. But this method is not as efficient as the servos would move at a constant rate regardless of how close or far the turret is to the target. Thus, to optimize movement of the servos, a PID control algorithm is implemented that would optimize the motor movement such that the desired target is reached faster.

The PID in a PID controller stands for proportional, integral, and derivative. Each part of the controller plays a role in controlling the motors such that the motors will not only move to the desired location quickly, but also with minimal oscillations. The proportional partsimply adjusts the speed/strength of movement proportional to the error. In other words, the further away the target is from the centre of the frame, the faster the motors will try and adjust. Similarly, the closer the target is to the centre, the slower the motors would adjust; the "strength" or speed of the motor is controlled by changing the size of the increment the motor moves at each time step.

While the proportional part of the controller is effective, only having the proportional part caused some oscillation when tracking; the servos sometimes overshot the desired position. So,the derivative part of the controller was incorporated into our system to tune down the change in velocity and dampen the oscillations.

c) Multi-processing algorithm:

The last part of the software design is the multi-processing algorithm. To minimize processing delay and allow the video to display with no perceivable delay, multiprocessing was essential.

A master process and three worker processes were designed to carry out necessary steps of the algorithms described above simultaneously. The master process performed steps 1-3 of the object recognition algorithm: it grabs a frame and checks if the frame has blue object before sending it to a queue in which one of the three worker processes will perform open/close operation and contour extraction; the three processes will rotate to grab frames from the queue and can operate in parallel, thus effectively decreasing overall computation time. Once processing is done for one frame, the master process will extract the contours sent back from the worker process, display the contours on the screen, and use the contours for PID servo control. This way, the master process grabs and displays the frame independently on the screen, allowing the video streaming from the camera to have no visual delay. Each of the three worker processes can process a frame in 150ms, and since processing of three frames is done at the same time now, on average, the processing time becomes around 50ms.

**4.3.3: Testing and implementation:**

Once the servo pan tilt kit was assembled, the servo motors were checked individually for working by running a code for complete 180 degree rotation (in steps of 0 degree, neutral 90 degree and 180 degree) from the RaspberryPi.

Since software generated PWM signals were too noisy and unstable to control the servos, generation of hardware PWM signals using pigpio library was implemented. So before the code is run, the command "taskset 0x1 sudopigpiod" is run in the terminal to run the code at the first core of the RPi. The hardware PWM generated (in GPIO pins 12 and 13) were much cleaner PWM signals and significantly reduced the amount of jitter created on the servos.

When the object recognition code was run, the error saying the following: *"OpenCV Error: Assertion failed (scn == 3 || scn == 4) in cvtColor, file /build/opencv-U1UwfN/opencv-2.4.9.1+dfsq1/modules/imgproc/src/color.cpp, line 3737"* was encountered. This was rectified by entering the command "sudomodprobe bcm2835-v4l2" in the terminal before execution of the code. This command basically installed the V42l driver on the Broadcom chip used in RPi B+.

Once the pan tilt and the object detection was working as desired, the two were merged and object detection was used to output feedback values to control the pan tilt with our PID control algorithm. Finally the laser was mounted on the tilt kit and a GPIO was used to toggle the laser whenever key “v” was pressed on the keyboard. The intention of this mechanism was to leave the final decision of firing of the turret to be left to the user instead of being automated.

A flow chart of the working of this module is given below-

****PTO**

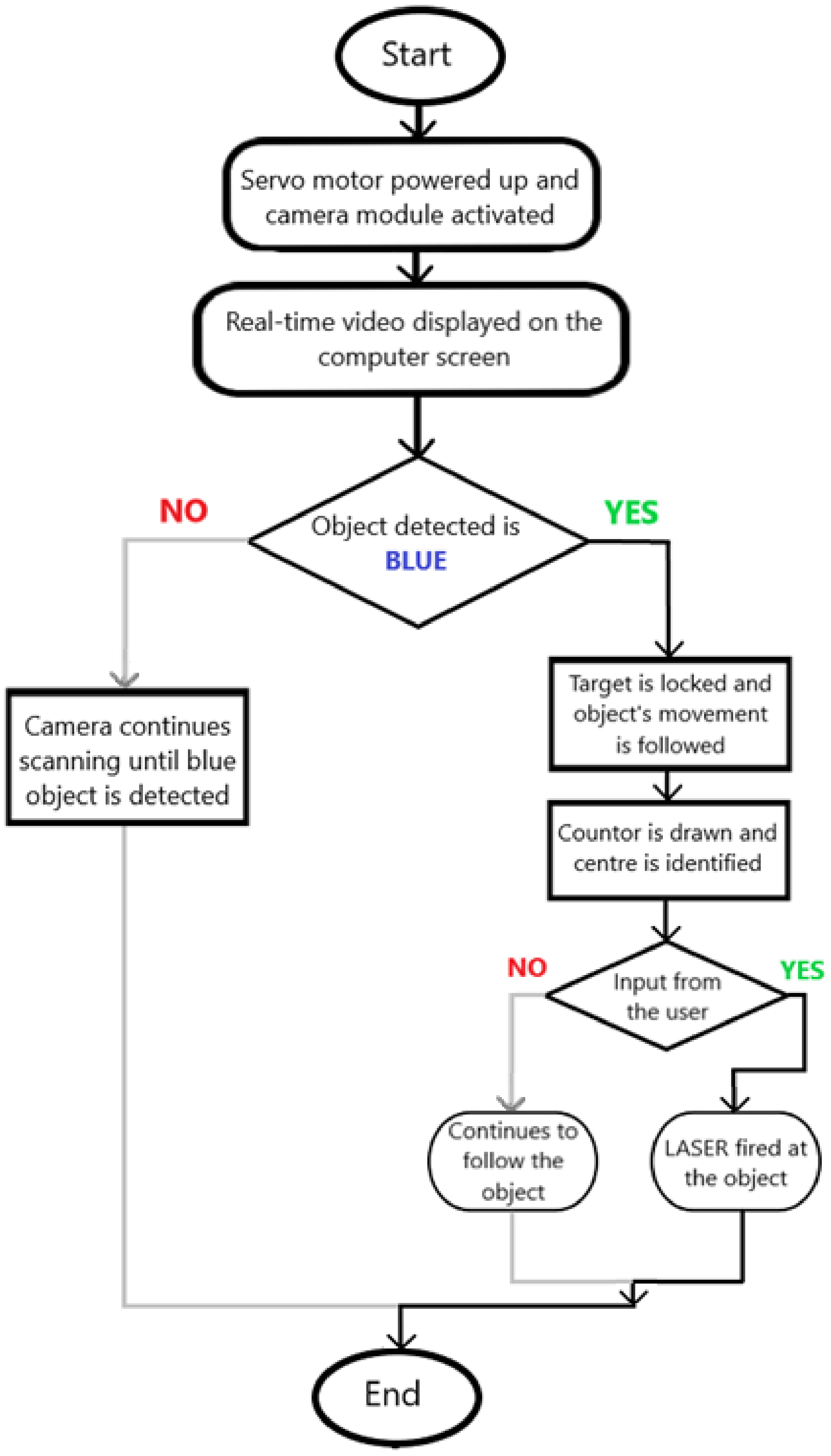

Start
Servo motor powered up and camera module activated
Real-time video displayed on the computer screen
Object detected is BLUE
NO
YES
Camera continues scanning until blue object is detected
Target is locked and object's movement is followed
Countor is drawn and centre is identified
Input from the user
NO
YES
Continues to follow the object
LASER fired at the object
End

**Figure 4.11: Flow Chart of Image Processing module.**

## 4.3.4)Working:

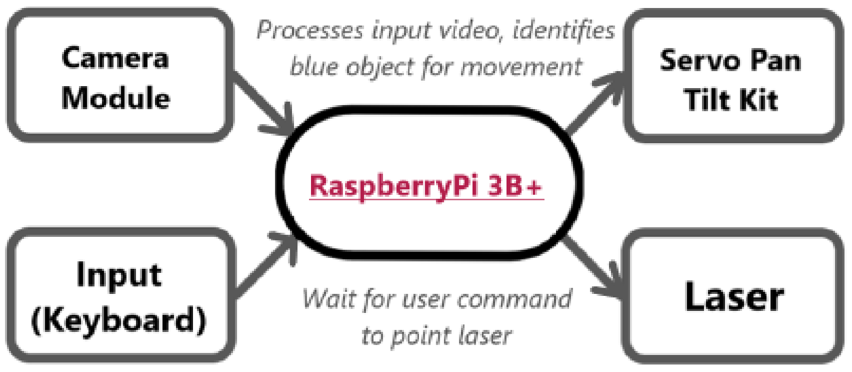

Camera Module
Processes input video, identifies blue object for movement
Servo Pan Tilt Kit
RaspberryPi 3B+
Input (Keyboard)
Wait for user command to point laser
Laser

**Figure 4.12: Block Diagram of Image Processing module**

Figure 4.5 shows the working of the Image Processing module. Its primary objective is to recognise a blue coloured object, target lock at its centre, track it as it moves (with the help of servos) and in the process display a real-time video to the user to take his/her decision to fire a LASER as the target.

The image processing code once developed is dumped into the RaspberryPi. Once the code is run and supply is given to the servos, the camera module gets activated causing a video to be displayed on the screen of the laptop. A blue object facing the camera and in its vicinity is recognised and a contour is drawn around it. The centre of the largest blue object is identified and target lock is established. So now, as the object moves the servos follow it continuously; one motor providing the left-right rotation and the other the up-down tilt. The servo motors moving causes the camera mounted on the tilt kit to also move such that the object is maintained at centre position.

A manual input from the keyboard allows the user to have the final say in the firing of the turret. By pressing the key “v” on the keyboard the laser is toggled to fire at the target. Since the laser is mounted on the tilt kit itself right next to the camera, its aim is pretty accurate. An additional feature is that by pressing the key “q” on the keyboard, all the processor cores stop execution and the model goes to sleep.

### 4.3.5) Autonomous LASER Turret forming the Target Acquisition module integrating object detection and image processing.

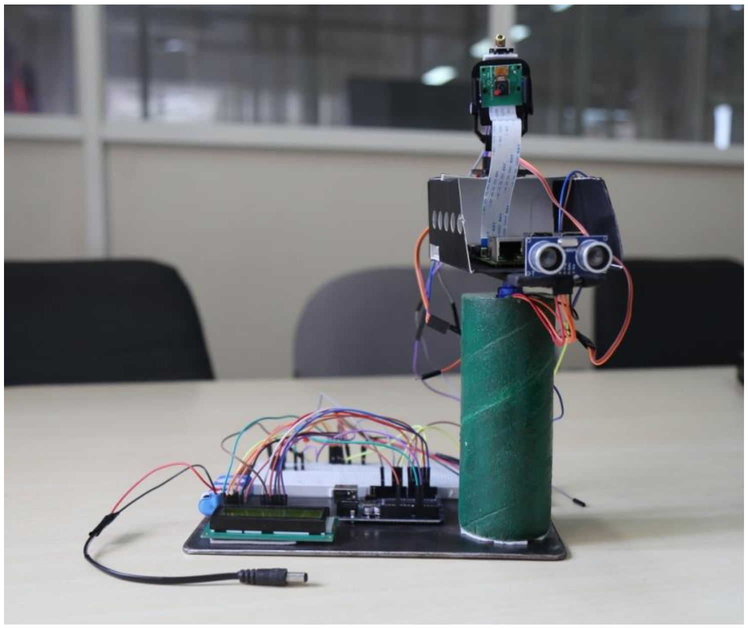


**Figure 4.13: Autonomous LASER Turret**

The model after combination of both the object detection and image processing modules resulting in the target acquisition part of the project is shown above. The turret detects the object, displays its exact position on the LCD screen and alerts the user about the object's presence. The real-time video is displayed on the computer screen which identifies a blue object, locks the target and follows its movement. An input from the user can neutralise the target using a LASER.

# 5. Results

**Preamble:-**

All terrain Semi-Automatic Rover with Autonomous Target Acquisition and GPS Trackingwas developed and executed. Once the design, development and testing were completed, the model was run, and the obtained results are discussed in this chapter.

## Execution of Image processing module

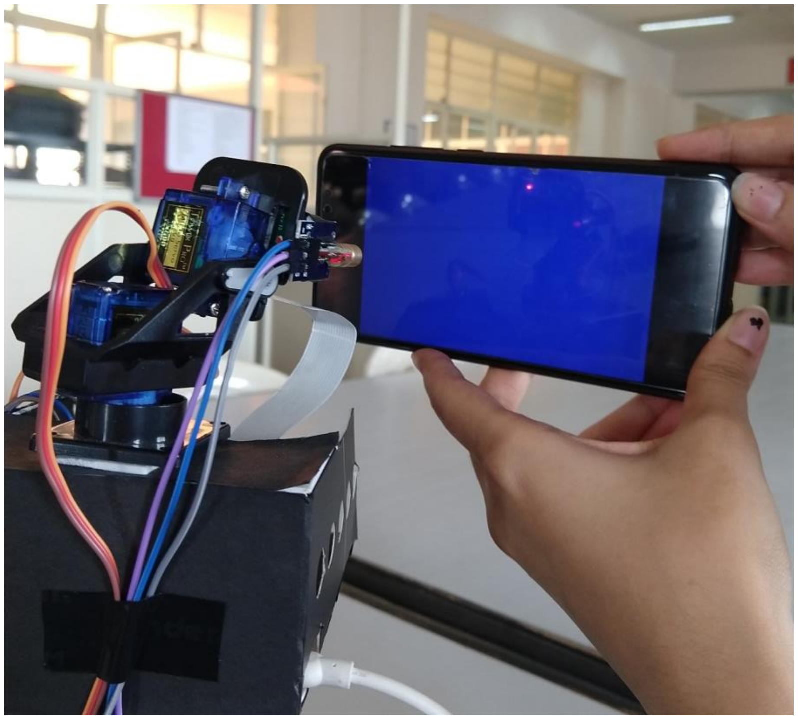


**Figure 5.1: Image Processing module target-locking the blue object**

Figure 5.1 shows the working of the image processing module. Here the object which is blue in colour is seen to be in the range of the turret. Then target is locked by the module and the LASER is being pointed towards it.

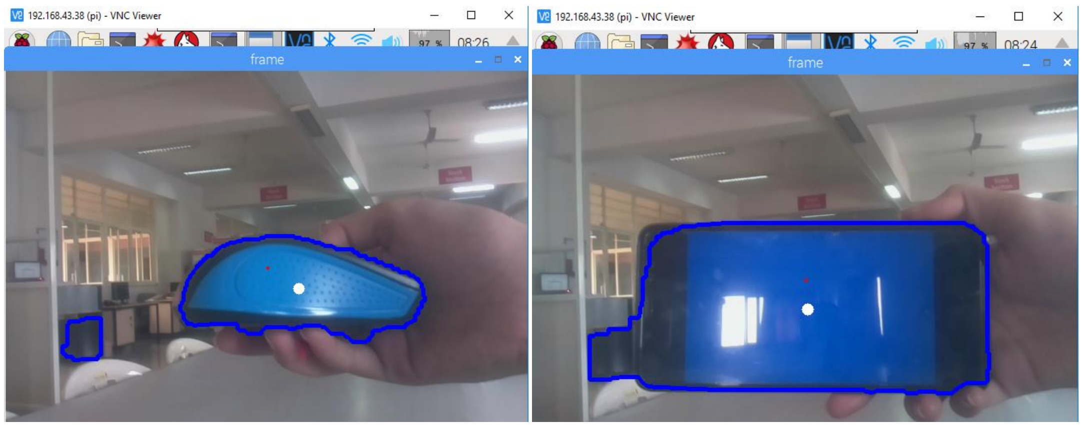
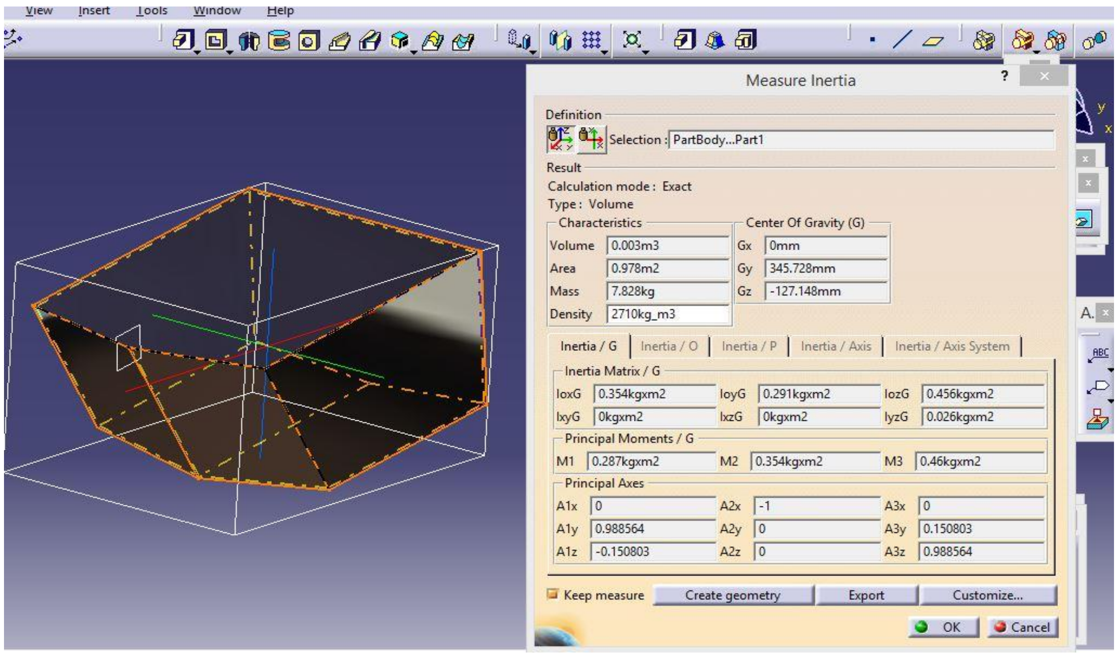

192.168.43.38 (pi) - VNC Viewer
frame
08:26
192.168.43.38 (pi) - VNC Viewer
frame
08:24

**Figure 5.2: Image Processing Module detecting blue objects of different shapes**

These images show the display that is obtained on the computer screen of the user. The RaspberryPi camera module records the object and transmits the video to the computer, which is connected to the RPi, hence enabling the user to live track the threat. There is contour drawn around the largest blue object that is seen by the camera and the its centre is indicated using a white dot. Also, the LASER is activated by the command of the user which points at the detected blue object and can be seen on the screen as well.

## Execution of Rover module



Definition

Selection : PartBody...Part1

Result

Calculation mode : Exact

Type : Volume

| Characteristics | | Center Of Gravity (G) | |
|---|---|---|---|
| Volume | 0.003m3 | Gx | 0mm |
| Area | 0.978m2 | Gy | 345.728mm |
| Mass | 7.828kg | Gz | -127.148mm |
| Density | 2710kg_m3 | | |

Inertia Matrix / G

| | | | | | |
|---|---|---|---|---|---|
| IoxG | 0.354kgxm2 | IoyG | 0.291kgxm2 | IozG | 0.456kgxm2 |
| IxyG | 0kgxm2 | IxzG | 0kgxm2 | IyzG | 0.026kgxm2 |

Principal Moments / G

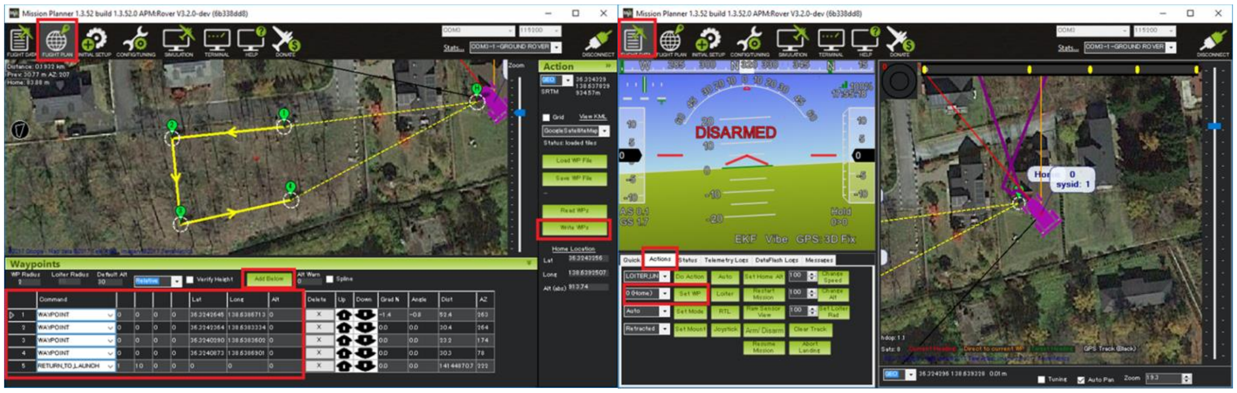
| | | | | | |
|---|---|---|---|---|---|
| M1 | 0.287kgxm2 | M2 | 0.354kgxm2 | M3 | 0.46kgxm2 |

Principal Axes

| | | | | | |
|---|---|---|---|---|---|
| A1x | 0 | A2x | -1 | A3x | 0 |
| A1y | 0.988564 | A2y | 0 | A3y | 0.150803 |
| A1z | -0.150803 | A2z | 0 | A3z | 0.988564 |

**Figure 5.3: Analysis of three dimensional rover model**

The overall shape and the dimensions of the rover model was obtained after performing the mass property analysis considering all the parameters. Satisfactory results were inferred from the said analysis performed for the hull of the rover.



WP Radius 2 | Loiter Radius | Default Alt 30 | Relative | Verify Height | Alt Warn 0 | Spline

| | Command | | | | | Lat | Long | Alt | Delete | Up | Down | Grad % | Angle | Dist | AZ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | WAYPOINT | 0 | 0 | 0 | 0 | 36.2242645 | 138.6386713 | 0 | X | | | -1.4 | -0.8 | 52.4 | 263 |
| 2 | WAYPOINT | 0 | 0 | 0 | 0 | 36.2242364 | 138.6383334 | 0 | X | | | 0.0 | 0.0 | 30.4 | 264 |
| 3 | WAYPOINT | 0 | 0 | 0 | 0 | 36.2240290 | 138.6383602 | 0 | X | | | 0.0 | 0.0 | 23.2 | 174 |
| 4 | WAYPOINT | 0 | 0 | 0 | 0 | 36.2240873 | 138.6386901 | 0 | X | | | 0.0 | 0.0 | 30.3 | 78 |
| 5 | RETURN_TO_LAUNCH | 1 | 10 | 0 | 0 | 0 | 0 | 0 | X | | | 0.0 | 0.0 | 141449707 | 222 |

**Figure 5.4: Rover autopilot simulation and waypoints.**

The image on the left describes the autonomous operation of the rover, it is calibrated and simulated on the mission planner software. The autonomous operation consists of the rover setting up its own course of direction based on the waypoints programmed.

The image on the right shows that in the Smart RTL function of the rover, it returns to the starting point in the shortest possible path.

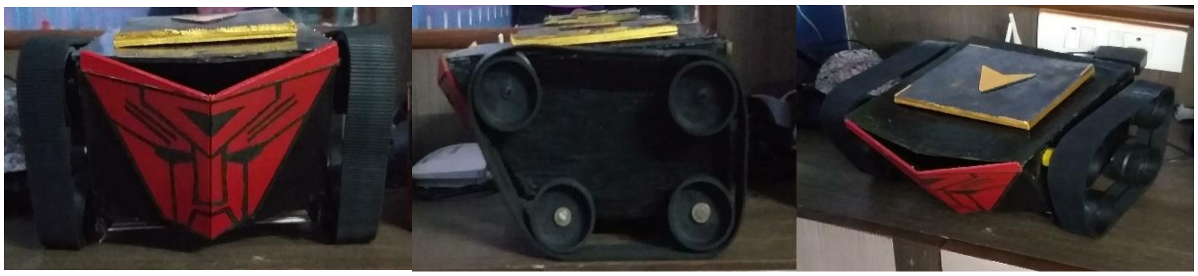


**Figure 5.5:A scaled (1:2) replica of the rover .**

# 6. Project Costing

Table 6.1 List of proposed components in this project and their cost:

| Components | Amount |
|---|---|
| Raspberry Pi | ₹2460 |
| Ultrasonic Sensor | ₹80 |
| Servo Motor (Nos. 3) | ₹270 |
| Arduino Uno | ₹450 |
| Raspberry Pi camera module | ₹620 |
| LASER | ₹120 |
| LCD Display | ₹125 |
| DC Supply Connector | ₹45 |
| LEDs (2 nos.) | ₹20 |
| Buzzer | ₹20 |
| Bread board | ₹80 |
| Jumper wires | ₹60 |
| DC Geared Motors | ₹1200 |
| Caterpillar Tracks | ₹900 |
| Hull(Body) Fabrication | ₹2000 |
| Pulley Wheels | ₹240 |

| | |
|---|---|
| Ublox GPS Module | ₹1400 |
| Electronic Speed Controller | ₹2400 |
| Telemetry Radio | ₹1800 |
| Lithium Polymer Battery | ₹4500 |
| GoPro Hero 7 Camera | ₹22,000 |
| Video Transmitter and Receiver | ₹4,500 |
| 3 axis Gimbal | ₹2000 |
| Radio Transmitter and Reciever | ₹6000 |
| Pixhawk PX4 Flight Controller | ₹9300 |
| Miscellaneous | ₹2000 |
| **Total Cost** | **₹64,590** |

# 7. Conclusions and Suggestions for Future Work

**Preamble:-**

The inferences drawn after developing this project, its real-world applications and the suggestions for future work are discussed in this final chapter.

## 7.1 Conclusion

- A prototype of the of an autonomous LASER turret for target acquisition was designed and built satisfactorily. The prototype of the target acquisition module was tested in a real-time scenario and was found to meet the objectives it was designed for. The addition of an ultrasonic sensor with a servo motor improved the view angle to 180 degrees. One of the advantages of this prototype is that it is portable. The operation of this prototype is fully automated till the target is locked and the real time video is accessible to the user. However, the user has the final say whether or not to fire the LASER at the object (which is analogous to neutralizing the target in this scenario).The designed model for object detection and tracking using Raspberry Pi and Arduino achieves its purpose by providing the expected results.The design of the rover model was simulated and the control unit of the rover was successfully studied.The global positioning system and telemetry radio is successfully studied and integrated as an attachment to the rover.Since the occurrence of the COVID-19 pandemic has affected the completion of the project, the practical scope of this project is limited to only the target acquisition module.

## 7.2 Applications

The real-world applications of this project are not limited to one domain. Due to its versatility, it can serve its purpose in multiple fields satisfying a wide range of objectives. Some of the applications are listed as follows:

1. In application of security, this rover can be utilised in selected areas of an establishment to periodically survey for any trespassers, unwanted guests and neutralise them if required using a high intensity laser.

2. On addition of a Sonar module, the rover can be made to detect and avoid obstacles.
3. With appropriate upgradation and application-specific modifications, this rover can be used in the defence forces of our country in places where human presence is not physically possible.
4. On addition of a thermal camera the rover`s surveillance capacity is enhanced because of thermally distinguishing the environment.
5. Apart from security applications,thisrover can be used fire rescue missions, on addition of a thermal camera .
6. With additional algorithms and image processing the rover can be made to follow any human being.

## 7.3 Suggestion for Future Development

This prototype was built on small scale and it can be further developed depending upon the need of the end user. It can be used for monitoring and safeguarding restricted areas. On further improvisation of the algorithm, the turret module can identify and differentiate two or more colours as a friend or a foe. Thesealgorithms and modifications can be complex and huge which can be executed as per the users' requirement.

The servo motors used have a rotation angle of 180 degrees which limits this prototype to have an operational range of only 180 degrees. To achieve a greater operational range, the servo motors can be replaced with stepper motors. However, stepper motors have greater power supply requirements,are less cost efficient and can make the model bulky.Instead, 2 servo motors can be attached back-to-back or multiple servo motors can be used(if space constraint isn't considered) to obtain complete 360 degree range for the turret.

# References

1. Newton, A. (2019). *Arduino RADAR Model using Ultrasonic Sensor for Detection & Ranging*. [online] How To Electronics. Available at: https://www.how2electronics.com/arduino-radar-model-ultrasonic-sensor/Accessed 29 Apr. 2019].
2. Kenny, T.G., Hamilton Thorne BioSciences Inc, 2008. *Microscope turret mounted laser EPI-illumination port*. U.S. Patent 7,359,116.
3. Francis Yu, Xitang Zhao, S., Trex Communications Corp, 2018. *Autonomous object tracking turret*. U.S. Patent 6,347,001.
4. Borenstein, J. and Koren, Y., 1988. Obstacle avoidance with ultrasonic sensors. *IEEE Journal on Robotics and Automation*, *4*(2), pp.213-218.
5. Hashimoto, H., Yamamoto, H., Yanagisawa, S. and Harashima, F., 1988. Brushless servo motor control using variable structure approach. *IEEE Transactions on industry applications*, *24*(1), pp.160-170.
6. Papageorgiou, C. and Poggio, T., 2000. A trainable system for object detection. *International journal of computer vision*, *38*(1), pp.15-33.
7. Farooq, W., Butt, N., Shukat, S., Baig, N. A., & Ahmed, S. M. (2016). Wirelessly Controlled Mines Detection Robot. 2016 International Conference on Intelligent Systems Engineering (ICISE).
8. Pavithra, S., and S. A. Siva San for military." "7TH sense-a multipurpose robot ommunication and Embedded Systems Conference on, pp. 1224-1228. IEEE, In Information Co (ICICES), 2013 International C 2013
9. Sandeep Bhat, Dr. M. Meenakshi," The Role of Wireless Communication for Autonomous Military Robot", Int'l Journal of Computing, Communications & Instrumentation Engg. (IJCCIE) Vol. 4, Issue 1 (2017) ISSN 2349-1469 EISSN 2349-1477.pp.64-66.
10. Zhenjun He, Jiang Zhang, Peng Xu,JiahengQin,andYunkai Zhu, "Mine detecting robot based on wireless communication with multi-sensor," IEEE 4th International Conference , vol no, pp.117- 120, 15-17 Nov. 2013.

11. U. Ashish and V. Ratnaparkhe, “Video Surveillance Robot Control using Smartphone and Raspberry Pi”, IEEE International Conference on Communication and Signal Processing, pp. 2094 – 2097, April, 2016.
12. Yu, Wenshuai, et al. "A new framework of moving targetdetection and tracking for UAV video application." (2008).
13. Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.
14. Kumar, Saurav, and Pallavi Awasthi. "Navigation architecture for autonomous surveillance rover." International Journal of Computer Theory and Engineering1.3 (2009).
15. Abdelhafid, B., et al. "Design and Implementation of an Unmanned Ground Vehicle for Security Applications." 7th International Symposium on Mechatronics and its Applications (ISMA10), Sharjah, UAE. 2010.
16. Iagnemma, Karl, et al. Design and development of an agile, man portable unmanned ground vehicle. MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF MECHANICAL ENGINEERING, 2008.
17. Berman, Sigal, Edna Schechtman, and Yael Edan. "Evaluation of automatic guided vehicle systems." Robotics and Computer-Integrated Manufacturing25.3 (2009): 522-528.
18. T. A. Salh and M. Z. Nayef, "Intelligent surveillance robot," *2013 International Conference on Electrical Communication, Computer, Power, and Control Engineering (ICECCPCE)*, Mosul, 2013, pp. 113-118, doi: 10.1109/ICECCPCE.2013.6998745.
19. http://en.wikipedia.org/wiki/DRDO Daksh.
20. H. Lang, Y. Wang and C. D. Silva, “Vision based object identification and tracking for mobile robot visual servo control,” in 8th IEEE International Conference on Control and Automation (ICCA), 2010.

# Appendix

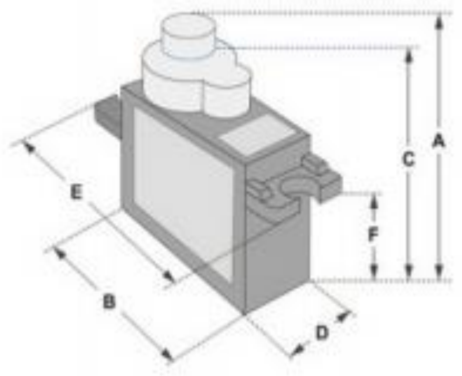
1. <u>RaspberryPi main code:</u>

```
defgrab_frame_display(run_flag, send_frame_queue, receive_contour_queue, p_start_turn, p_end_turn, p_start_lock, p_end_lock):
 local =True
 last_contour_receive_time=0
 startTime_ms=0
 contourRead=False
 x_diff=0
 y_diff=0
 prevX_diff=0
        prevY_diff=0
        start_time=0
        start_datetime=datetime.now()
        laserCond=False
        calibrationNow=False
        while (run_flag.value):
                # 1. Extract a frame from the video
                returnBoolean, frame =videoCap.read()
                # 2. Convert RGB color space to HSV (hue, saturation, value) color space
                frame_hsv= cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
                # Define range of blue color in HSV
                # For HSV, Hue range is [0,179], Saturation range is [0,255] and Value range is [0,255].
                # H maps to 0-360 degree and S, V maps to 0-100%
                # 3. Threshold the HSV image to get only blue colors
                mask = cv2.inRange(frame_hsv, lower_blue, upper_blue)
                # 4. Check the sum of blue pixels, only try to send over for processing if greater than threshold
                sum_of_blue=np.sum(mask)
                exceed_threshold=sum_of_blue>blue_threshold
                # 5. Check if time since last send to queue exceeds 30ms
                current_time=datetime.now()
                delta_time=current_time-start_datetime
                delta_time_ms=delta_time.total_seconds()*1000
                # Check if at Calibration Mode
                if (calibrationNow):
                        pi_hw.hardware_PWM(motorPinR, 50, 80000) #50 Hz Freq.
                        pi_hw.hardware_PWM(motorPinT, 50, 75000) #50 Hz Freq.
                else:
                        # Only put frame in queue if it has past 30ms and exceeds blue threshold and there are fewer than 4 frames in queue
                        if ((delta_time_ms>30) andexceed_thresholdand (send_frame_queue.qsize() <4)):
```

```
                    start_datetime=current_time# Update last send to queue time
                    send_frame_queue.put(mask)  # Put mask in queue
                #Check if receive_contour_queue is not empty
                if ((notreceive_contour_queue.empty())):
                    last_contour_receive_time=time.time()
                    contours =receive_contour_queue.get()  #Extract contour
                    # Extract the contour of the largest blue object
                    if (len(contours)>0):
                        maxContour=len(contours)
                        maxIndex=0
                    foriinrange (len(contours)):
                        if (len(contours[i]) >maxContour):
                            maxContour=len(contours[i])
                            maxIndex=i
                        M = cv2.moments(contours[maxIndex])
                    # Compute center position of the largest blue object
                    if (M["m00"] !=0):
                        cX=int(M["m10"] / M["m00"])
                        cY=int(M["m01"] / M["m00"])
                        # PID Control Algo to calculate strength for servo control
                        x_diff=abs(cX-center_x)
                        y_diff=abs(cY-center_y)
                        kp_x=3
                        kd_x=0.005
                        kp_y=3
                        kd_y=0.005
                        proportional_x=x_diff/(x_res/2.0)
                        proportional_y=y_diff/(y_res/2.0)
                        derivative_x= (prevX_diff-x_diff)/(time.time() -start_time)
                        derivative_y= (prevY_diff-y_diff)/(time.time() -start_time)
                        #print("derivative_x: " + str(derivative_x))
                        #print("derivative_x*kd_x: " + str(derivative_x*kd_x))
                        start_time=time.time()
                        strength_x=proportional_x*kp_x-derivative_x*kd_x
                        strength_y=proportional_y*kp_y-derivative_y*kd_y
                        #print "strength:"
                        #print strength_x

                    # Check if within range, move servos if not
                    if (x_diff<=x_tol): #Within range, do nothing
                        a =1
                        #print("horizontal-axis in range")
                    elif (cX>center_x):
                        #print("Move Right by ", x_diff, "px")
                        rotate5(1,strength_x)
                    else:
                        #print("Move Left by ", x_diff, "px")
                        rotate5(-1,strength_x)
                    if (y_diff<=y_tol): #Within range, do nothing
                        a =1
                        #print("vertical-axis in range")
                    elif (cY>center_y):
                        #print("Move Down by ", y_diff, "px")
                        tilt5(-1,strength_y)
                    else:
                        #print("Move Up by ", y_diff, "px")
                        tilt5(1,strength_y)
                    #print("--------")
                    prevX_diff=x_diff
                    prevY_diff=y_diff
            #Display last receiving contours for 0.5 sec
                if ((time.time()-last_contour_receive_time)<0.5):
                    cv2.circle(frame, (cX, cY), 7, (255, 255, 255), -1) #Draw center of object
                    cv2.drawContours(frame,contours,-1,(255,0,0),3) #Draw contour of object

                cv2.circle(frame, (center_x, center_y), 2, (0, 0, 255), -1) #Draw center of camera
                cv2.imshow('frame',frame) #Display Frame

                k = cv2.waitKey(5) &0xFF
                if k ==ord('q'): # Press q to exit program safely
                    run_flag.value=0
                    print("set run_flag --- 0")
                elif k ==ord('b'): # Press b to fire rubber band
                    tilt5(1, 6)
                    fire()
                    print("")
                elif k ==ord('v'): # Press v to toggle laser
                    laserCond=notlaserCond
                    iflaserCond:
                        print("Laser On")
                    else:
                        print("Laser Off")
                    laser(laserCond)
                elif k ==ord('m'): # Press m to enter/exit calibration mode to launch runnber band
                    calibrationNow=notcalibrationNow
                    ifcalibrationNow:
                        print ("Enter calibration mode")
                    else:
                        print ("Exit calibration mode")
```

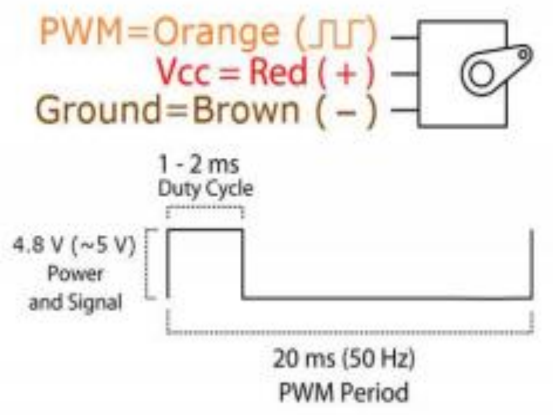
## SERVO MOTOR SG90 DATA SHEET

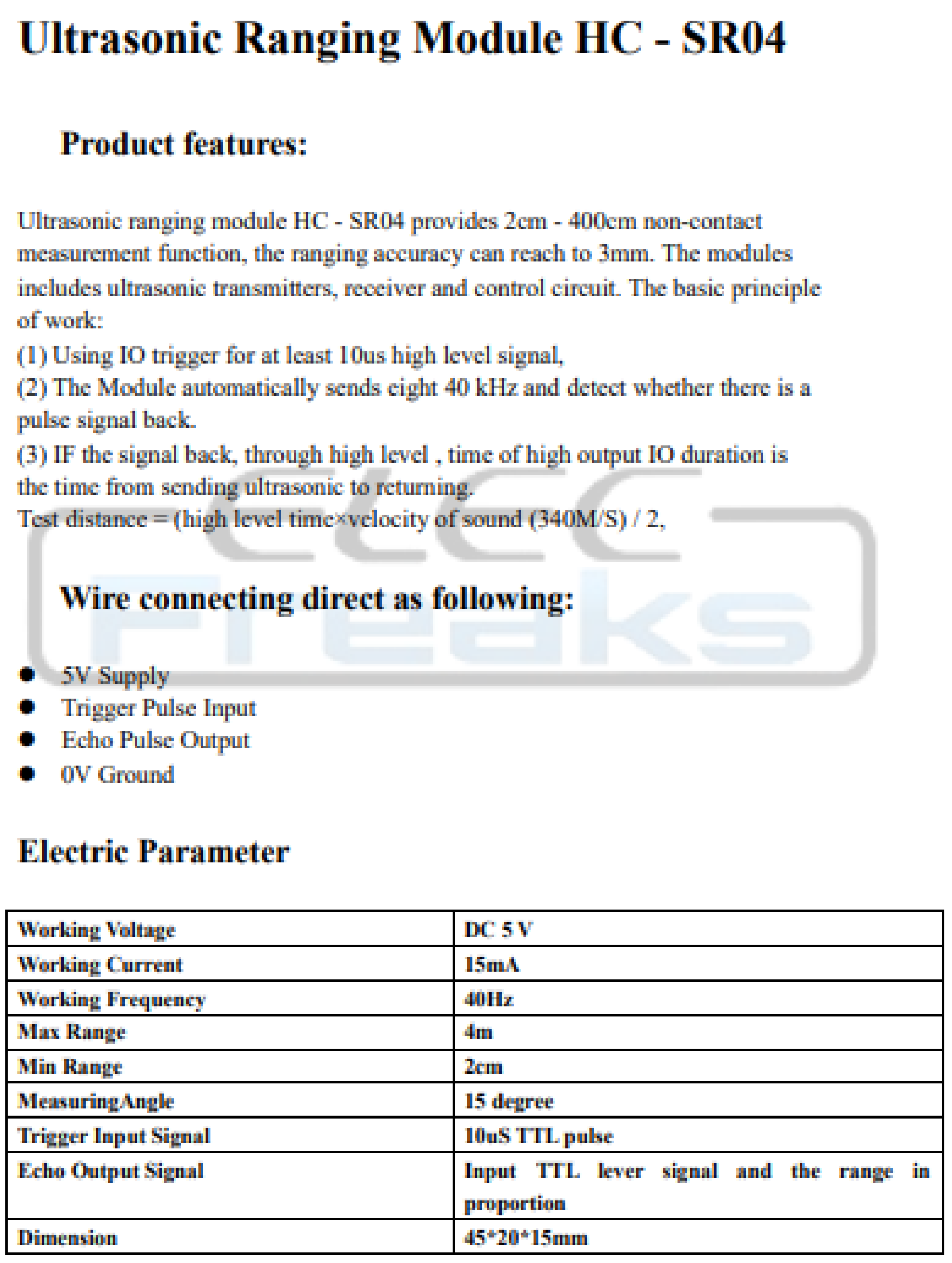
Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



| Dimensions & Specifications |
|---|
| A (mm) : 32 |
| B (mm) : 23 |
| C (mm) : 28.5 |
| D (mm) : 12 |
| E (mm) : 32 |
| F (mm) : 19.5 |
| Speed (sec) : 0.1 |
| Torque (kg-cm) : 2.5 |
| Weight (g) : 14.7 |
| Voltage : 4.8 - 6 |

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.





## Ultrasonic Ranging Module HC - SR04

### Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

### Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

### Electric Parameter

| | |
|---|---|
| **Working Voltage** | **DC 5 V** |
| **Working Current** | **15mA** |
| **Working Frequency** | **40Hz** |
| **Max Range** | **4m** |
| **Min Range** | **2cm** |
| **MeasuringAngle** | **15 degree** |
| **Trigger Input Signal** | **10uS TTL pulse** |
| **Echo Output Signal** | **Input TTL lever signal and the range in proportion** |
| **Dimension** | **45*20*15mm** |